



ΠΑΝΕΠΙΣΤΗΜΙΟ
ΘΕΣΣΑΛΙΑΣ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΥΒΡΙΔΙΚΗ ΕΦΑΡΜΟΓΗ ΑΝΑΖΗΤΗΣΗΣ ΘΕΣΕΩΝ ΣΤΑΘΜΕΥΣΗΣ

ΚΩΝΣΤΑΝΤΙΝΟΣ ΣΠΑΝΔΩΝΙΔΗΣ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΥΠΕΥΘΥΝΟΣ

ΤΖΙΑΛΛΑΣ ΓΡΗΓΟΡΙΟΣ

Καθηγητής



ΠΑΝΕΠΙΣΤΗΜΙΟ
ΘΕΣΣΑΛΙΑΣ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΥΒΡΙΔΙΚΗ ΕΦΑΡΜΟΓΗ ΑΝΑΖΗΤΗΣΗΣ ΘΕΣΕΩΝ ΣΤΑΘΜΕΥΣΗΣ

ΚΩΝΣΤΑΝΤΙΝΟΣ ΣΠΑΝΔΩΝΙΔΗΣ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΥΠΕΥΘΥΝΟΣ

ΤΖΙΑΛΛΑΣ ΓΡΗΓΟΡΙΟΣ

Καθηγητής

Λαμία, 2023



UNIVERSITY OF
THESSALY

SCHOOL OF SCIENCE

DEPARTMENT OF COMPUTER SCIENCE & TELECOMMUNICATIONS

HYBRID APPLICATION FOR SEARCHING FREE PARKING SPOTS

KONSTANTINOS SPANDONIDIS

FINAL THESIS

ADVISOR

TZIALLAS GRIGORIOS

Professor

Lamia, 2023

«Με ατομική μου ευθύνη και γνωρίζοντας τις κυρώσεις ⁽¹⁾, που προβλέπονται από της διατάξεις της παρ. 6 του άρθρου 22 του Ν. 1599/1986, δηλώνω ότι:

1. Δεν παραθέτω κομμάτια βιβλίων ή άρθρων ή εργασιών άλλων αυτολεξεί **χωρίς να τα περικλείω σε εισαγωγικά** και χωρίς να αναφέρω το συγγραφέα, τη χρονολογία, τη σελίδα. Η αυτολεξεί παράθεση χωρίς εισαγωγικά χωρίς αναφορά στην πηγή, είναι λογοκλοπή. Πέραν της αυτολεξεί παράθεσης, λογοκλοπή θεωρείται και η παράφραση εδαφίων από έργα άλλων, συμπεριλαμβανομένων και έργων συμφοιτητών μου, καθώς και η παράθεση στοιχείων που άλλοι συνέλεξαν ή επεξεργάστηκαν, χωρίς αναφορά στην πηγή. Αναφέρω πάντοτε με πληρότητα την πηγή κάτω από τον πίνακα ή σχέδιο, όπως στα παραθέματα.

2. Δέχομαι ότι η αυτολεξεί **παράθεση χωρίς εισαγωγικά**, ακόμα κι αν συνοδεύεται από αναφορά στην πηγή σε κάποιο άλλο σημείο του κειμένου ή στο τέλος του, είναι αντιγραφή. Η αναφορά στην πηγή στο τέλος π.χ. μιας παραγράφου ή μιας σελίδας, δεν δικαιολογεί συρραφή εδαφίων έργου άλλου συγγραφέα, έστω και παραφρασμένων, και παρουσίασή τους ως δική μου εργασία.

3. Δέχομαι ότι υπάρχει επίσης περιορισμός στο μέγεθος και στη συχνότητα των παραθεμάτων που μπορώ να εντάξω στην εργασία μου εντός εισαγωγικών. Κάθε μεγάλο παράθεμα (π.χ. σε πίνακα ή πλαίσιο, κλπ), προϋποθέτει ειδικές ρυθμίσεις, και όταν δημοσιεύεται προϋποθέτει την άδεια του συγγραφέα ή του εκδότη. Το ίδιο και οι πίνακες και τα σχέδια

4. Δέχομαι όλες τις συνέπειες σε περίπτωση λογοκλοπής ή αντιγραφής.

Ημερομηνία:/...../20.....

Ο – Η Δηλ.

ΣΠΑΝΔΩΝΙΔΗΣ ΚΩΝΣΤΑΝΤΙΝΟΣ

(1) «Όποιος εν γνώσει του δηλώνει ψευδή γεγονότα ή αρνείται ή αποκρύπτει τα αληθινά με έγγραφη υπεύθυνη δήλωση

του άρθρου 8 παρ. 4 Ν. 1599/1986 τιμωρείται με φυλάκιση τουλάχιστον τριών μηνών. Εάν ο υπαίτιος αυτών των πράξεων

σκόπευε να προσπορίσει στον εαυτόν του ή σε άλλον περιουσιακό όφελος βλάπτοντας τρίτον ή σκόπευε να βλάψει άλλον, τιμωρείται με κάθειρξη μέχρι 10 ετών.»

ΕΥΧΑΡΙΣΤΙΕΣ

Με την ολοκλήρωση της διπλωματικής μου εργασίας θα ήθελα ευχαριστήσω τα άτομα που με βοήθησαν, χωρίς τη βοήθεια των οποίων πιθανόν να μην ήταν δυνατή η εκπόνησή τη. Αρχικά, ευχαριστώ τον καθηγητή μου κ. Γρηγόριο Τζιάλλα για τη επίβλεψη της εργασίας, τη συμπαράσταση και τη δυνατότητα που μου έδωσε να ασχοληθώ με ένα θέμα που με ενδιέφερε πολύ. Επίσης θα ήθελα να ευχαριστήσω όλους μου τους φίλους μου και την αδερφή μου, διότι στάθηκαν δίπλα μου καθ' όλη την διάρκεια της εκπόνησης της διπλωματικής μου. Τέλος, θα ήθελα ευχαριστήσω τους γονείς μου για όλες τις θυσίες που έκαναν ώστε να καταφέρω να τελειώσω αυτή τη διπλωματική αλλά και την σχολή μου.

Η διπλωματική εργασία επικεντρώνεται στο σημαντικό πρόβλημα της στάθμευσης στα αστικά περιβάλλοντα και παρουσιάζει μια καινοτόμο λύση για τη βελτιστοποίηση της χρήσης των διαθέσιμων χώρων στάθμευσης. Το πρόβλημα αυτό υπάρχει σε πολλές πόλεις παγκοσμίως, καθώς η αυξανόμενη χρήση αυτοκινήτων έχει ως αποτέλεσμα την υπερβολική ζήτηση για χώρους στάθμευσης, επηρεάζοντας αρνητικά την ποιότητα ζωής των πολιτών.

Στη διπλωματική εργασία παρουσιάζεται η ιδέα της δημιουργίας μιας υβριδικής εφαρμογής για την ανταλλαγή θέσεων στάθμευσης. Η εφαρμογή αυτή, που βασίζεται σε τεχνολογικά πλαίσια όπως το Laravel, το Quasar και το electron.js, έχει ως στόχο να προσφέρει λύση στη διαχείριση της ζήτησης στάθμευσης. Αντί να επικεντρώνεται απλώς στην εύρεση θέσεων, η εφαρμογή επιτρέπει στους χρήστες να ανταλλάσσουν θέσεις στάθμευσης, βελτιστοποιώντας τη χρήση των ήδη υπαρχόντων θέσεων.

Η προτεινόμενη λύση προσφέρει πολλαπλά οφέλη. Πρώτον, είναι ευέλικτη καθώς επιτρέπει την ανταλλαγή θέσεων σε οποιαδήποτε τοποθεσία, ενισχύοντας την ευκολία χρήσης. Δεύτερον, είναι οικονομικά αποδοτική αφού λειτουργεί ως δωρεάν υπηρεσία, αντίθετα με πολλές υπάρχουσες εφαρμογές που επιβάλλουν χρεώσεις. Τρίτον, συμβάλλει στη βιωσιμότητα, καθώς μειώνει τον χρόνο που απαιτείται για την εύρεση θέσεων, ελαχιστοποιώντας την κυκλοφοριακή συμφόρηση και την ατμοσφαιρική ρύπανση.

ABSTRACT

This thesis focuses on the importance of the problem of parking in urban environments and presents an innovative solution to optimize the use of available parking spaces. This problem exists in many cities worldwide, as the increasing use of cars results in excessive demand for parking spaces, negatively affecting the quality of life of citizens.

This thesis presents the idea of creating a hybrid application for the exchange of parking spaces. This application, based on technological frameworks such as Laravel, Quasar and electron.js, aims to provide a solution to parking demand management. Rather than just focusing on finding spaces, the app allows users to swap parking spaces, optimizing the use of existing spaces.

The proposed solution offers multiple benefits. First, it is flexible as it allows the exchange of seats in any location, enhancing ease of use. Second, it is cost-effective since it operates as a free service, unlike many existing apps that charge fees. Third, it contributes to sustainability by reducing the time needed to find seats, minimizing traffic congestion and air pollution.

Περιεχόμενα

ΠΕΡΙΛΗΨΗ	9
ABSTRACT	10
ΚΕΦΑΛΑΙΟ 1 Εισαγωγή	3
Το πρόβλημα της στάθμευσης 1.1	3
Στόχος 1.2	3
ΚΕΦΑΛΑΙΟ 2 Τύποι Εφαρμογών	5
Εγγενείς Εφαρμογές (Native apps) 2.1	5
Εφαρμογές Ιστού (Web apps) 2.2	5
Υβριδικές Διαδικτυακές Εφαρμογές - Hybrid Web Apps 2.3	6
Τα πλεονεκτήματα 2.3	7
ΚΕΦΑΛΑΙΟ 3 Περιγραφή του περιβάλλοντος ανάπτυξης	8
Server-Side Programming (Backend) 3.1	8
MySQL 3.1.1	8
PHP 3.1.2	9
Laravel PHP Framework 3.1.3	10
JavaScript Object Notation 3.1.4	10
API 3.1.5	12
Representational State Transfer (Rest API) 3.1.6	13
Axios Api 3.1.7	13
Client-Side Programming (Frontend) 3.2	14
HTML 3.2.1	14
JavaScript 3.2.2	14
Cascading Style Sheets (CSS) 3.2.3	14

Vue.js Πλαίσιο της JavaScript 3.2.4.....	15
Pinia 3.2.5	16
Quasar Πλαίσιο της JavaScript 3.2.6.....	16
Electron.js 3.2.7	17
ΚΕΦΑΛΑΙΟ 4 Περιγραφή της εφαρμογής σχεδιασμός και ανάπτυξη της βάσης δεδομένων.....	18
Εισαγωγή 4.1.....	18
Σχεδίαση βάσης δεδομένων 4.2.....	18
Βήματα δημιουργίας βάσης και πινάκων με την Laravel 4.2.1.....	19
Ο πίνακας users 4.2.2	21
Ο πίνακας parking_spots 4.2.3	21
Ο πίνακας reservations 4.2.4	22
Περιγραφή της εφαρμογής 4.3	22
Landing page	22
Login page.....	23
Register Page	24
Home Page	24
Search for Spot.....	25
ΚΕΦΑΛΑΙΟ 5 Συμπεράσματα	34
Προτάσεις για μελλοντική ανάπτυξη.....	35
Βιβλιογραφία	35
Βιβλία	35
Ηλεκτρονικές Πηγές.....	36
Παράρτημα-Πηγαίος Κώδικας.....	36
Παράρτημα Α	36
Παράρτημα Β Πηγαίος Κώδικας.....	37

ΚΕΦΑΛΑΙΟ 1 Εισαγωγή

Το πρόβλημα της στάθμευσης 1.1

Η στάθμευση είναι άμεση συνέπεια της μετακίνησης, είτε αφορά τον ανθρώπινο πληθυσμό είτε τη μεταφορά εμπορευμάτων. Η σημασία της είναι εμφανής τόσο στους διαθέσιμους χωρικούς πόρους όσο και στον χρόνο που επενδύεται για τη στάθμευση ενός οχήματος σε σχέση με τον ενεργό χρόνο οδήγησης. Κάθε ιδιωτικό όχημα απαιτεί την παρουσία ενός προσβάσιμου σημείου στάθμευσης κοντά στην κατοικία του ιδιοκτήτη του και τη δυνατότητα εντοπισμού στάθμευσης κοντά στο χώρο εργασίας του ή σε οποιονδήποτε άλλο προορισμό που απαιτεί ταξίδι. Όσον αφορά τη διάρκεια κατά την οποία ένα μέσο αυτοκίνητο παραμένει ακίνητο, με βάση θεωρητικά δεδομένα, υπολογίζεται ότι καλύπτει πάνω από το 90% του συνολικού χρόνου του, χρησιμοποιώντας πολλές θέσεις στάθμευσης καθημερινά ελλείψει ιδιωτικού χώρου στάθμευσης. Τα τελευταία χρόνια, τα κέντρα των πόλεων αντιμετωπίζουν σοβαρά προβλήματα λόγω της παρουσίας αυτοκινήτων, δημιουργώντας ένα επείγον θέμα σχετικά με την εύρεση χώρου στάθμευσης. Οι περιοχές κατά μήκος των δρόμων, γεμάτες από σταθμευμένα αυτοκίνητα, σε συνδυασμό με την κυκλοφορία αυτοκινήτων που αναζητούν χώρο στάθμευσης, επηρεάζουν σημαντικά την ποιότητα ζωής των κατοίκων και των εργαζομένων. Αυτό το πρόβλημα έχει επίσης οικονομικές συνέπειες.

Το πρόβλημα της στάθμευσης επηρεάζεται από αρκετούς παράγοντες. Παρακάτω βλέπουμε κάποιους από αυτούς[1]:

- Ανεπαρκής ποσότητα χώρων στάθμευσης
- Λανθασμένη εκμετάλλευση των χώρων στάθμευσης. Συγκεκριμένα κάποιοι χώροι στάθμευσης υπολειτουργούν, λόγω της κακής χωροταξίας ή λόγω του υψηλού κόστους που απαιτείται για στάθμευση.
- Η λανθασμένη χωροταξική διάταξη των θέσεων στάθμευσης σχετίζεται και με τον σχεδιασμό ενός χώρου στάθμευσης.
- Αδιαφορία του κράτους και της κοινωνίας για ύπαρξη ή δημιουργία εναλλακτικών τρόπων μεταφοράς που θα μειώσουν το πρόβλημα κυκλοφορίας.

Στόχος 1.2

Στόχος σε αυτή τη διπλωματική εργασία είναι η πρόταση μιας εναλλακτικής λύσης στο πρόβλημα της στάθμευσης. Η λύση που προτείνεται είναι η δημιουργία μιας υβριδικής εφαρμογής για την εύρεση θέσεων στάθμευσης. Με την χρήση της Laravel framework για την διαχείριση του back-end, της Quasar για το front-end και της electron.js για να γίνει η εφαρμογή υβριδική.

- Είναι πιο ευέλικτη: Άλλες εφαρμογές στάθμευσης συνήθως επιτρέπουν στους χρήστες να βρίσκουν και να κάνουν κράτηση θέσεων στάθμευσης σε συγκεκριμένες τοποθεσίες. Μια εφαρμογή για ανταλλαγή θέσεων στάθμευσης επιτρέπει στους χρήστες να ανταλλάσσουν θέσεις στάθμευσης με άλλους χρήστες σε οποιαδήποτε τοποθεσία. Αυτό καθιστά την εφαρμογή πιο ευέλικτη και επιτρέπει στους χρήστες να βρίσκουν σημεία στάθμευσης που είναι πιο κοντά στον προορισμό τους ή που είναι πιο βολικά για αυτούς.
- Είναι πιο οικονομική: Άλλες εφαρμογές στάθμευσης χρεώνουν συχνά στους χρήστες ένα ποσό για την εύρεση και κράτηση θέσεων στάθμευσης. Μια εφαρμογή για ανταλλαγή θέσεων στάθμευσης είναι συνήθως δωρεάν και οι χρήστες μπορούν να ανταλλάξουν θέσεις στάθμευσης μεταξύ τους.
- Είναι πιο βιώσιμη: Άλλες εφαρμογές στάθμευσης συχνά συμβάλλουν στην κυκλοφοριακή συμφόρηση και στην ατμοσφαιρική ρύπανση. Μια εφαρμογή για την ανταλλαγή θέσεων στάθμευσης συμβάλλει στη μείωση του χρόνου που περνούν οι οδηγοί κάνοντας κύκλους αναζητώντας θέσεις στάθμευσης. Αυτό μπορεί να βοηθήσει στη μείωση της κυκλοφοριακής συμφόρησης και της ατμοσφαιρικής ρύπανσης.

Οι κύριες διαφορές της πρότασης που παρουσιάζεται στην παρούσα εργασία σε σύγκριση με τις ήδη υπάρχουσες επιλογές είναι:

- Οι εφαρμογές στάθμευσης συνήθως επιτρέπουν στους χρήστες να βρίσκουν και να κάνουν κράτηση θέσεων στάθμευσης, σε δημόσιους ή ιδιωτικούς χώρους στάθμευσης, που συνήθως λειτουργούν επί πληρωμή. Μια εφαρμογή για ανταλλαγή θέσεων στάθμευσης επιτρέπει στους χρήστες να ανταλλάσσουν θέσεις στάθμευσης σε οποιαδήποτε τοποθεσία χωρίς οικονομική επιβάρυνση.
- Άλλες εφαρμογές στάθμευσης συχνά εστιάζουν στην εύρεση θέσεων στάθμευσης σε δημοφιλείς περιοχές, όπως περιοχές στο κέντρο της πόλης, κοντά σε τουριστικά αξιοθέατα και κοντά σε αεροδρόμια. Μια εφαρμογή όπως αυτή που περιγράφουμε μπορεί να χρησιμοποιηθεί για την εύρεση θέσεων στάθμευσης σε οποιαδήποτε περιοχή, συμπεριλαμβανομένων των κατοικημένων περιοχών εκτός του κέντρου της πόλης και των λιγότερο δημοφιλών εμπορικών περιοχών.

ΚΕΦΑΛΑΙΟ 2 Τύποι Εφαρμογών

Εγγενείς Εφαρμογές (Native apps) 2.1

Οι εγγενείς εφαρμογές (Native apps) είναι εφαρμογές λογισμικού που έχουν αναπτυχθεί για να λειτουργούν αποκλειστικά σε προκαθορισμένο λειτουργικό σύστημα. Ένα καθοριστικό χαρακτηριστικό των γνήσιων εγγενών εφαρμογών είναι η ανεμπόδιστη πρόσβασή τους στο υποκείμενο υλικό της συσκευής, επιπλέον της απρόσκοπτης ενσωμάτωσής τους με όλα τα στοιχεία διεπαφής πελάτη και τις διαδραστικές λειτουργίες που είναι εγγενείς στο συγκεκριμένο περιβάλλον λειτουργίας, χρησιμοποιώντας τη γλώσσα προγραμματισμού που υποστηρίζεται εγγενώς από αυτό το λειτουργικό σύστημα. Αυτή η αποκλειστική σχεδιαστική προσέγγιση διασφαλίζει τη βέλτιστη συμβατότητα και απόδοση εντός του καθορισμένου κινητού οικοσυστήματος.

Οι εγγενείς εφαρμογές συχνά αντιπαρατίθενται με εφαρμογές που βασίζονται στο Web που φιλοξενούνται σε διακομιστές και εκτελούνται σε προγράμματα περιήγησης Ιστού. Οι εφαρμογές Ιστού, που βασίζονται σε πρόγραμμα περιήγησης, ερμηνεύονται από προγράμματα περιήγησης μέσω JavaScript και HTML, ανεξάρτητα από την πλατφόρμα στην οποία λειτουργεί το πρόγραμμα περιήγησης, καθιστώντας τις καθολικά προσβάσιμες.

Αντίθετα, οι εγγενείς εφαρμογές παρουσιάζουν σταθερά ανώτερη απόδοση σε σύγκριση με τις αντίστοιχες εφαρμογές Web. Αυτό το πλεονέκτημα απόδοσης πηγάζει από την απουσία της διαδικασίας μετάφρασης μεταξύ του πηγαίου κώδικα μιας ιστοσελίδας και της γλώσσας μηχανής του υπολογιστή. Ωστόσο, η αισθητή διαφορά στην ταχύτητα που βιώνουν οι χρήστες μπορεί να ποικίλλει ανάλογα με την έκταση της υπολογιστικής επεξεργασίας που είναι εγγενής στην εφαρμογή.

Εφαρμογές Ιστού (Web apps) 2.2

Μια εφαρμογή Ιστού αναφέρεται σε μια εφαρμογή στην οποία έχει πρόσβαση μέσω ενός προγράμματος περιήγησης στο Διαδίκτυο. Οι πρόσφατες εξελίξεις στις τεχνολογίες Ιστού, τις γλώσσες προγραμματισμού και τις μεθοδολογίες έχουν ανοίξει την πόρτα στη δημιουργία δυναμικών εφαρμογών που εισάγουν ένα νέο μοντέλο συνεργασίας μεταξύ μιας μεγάλης βάσης χρηστών. Η ανάπτυξη διαδικτυακών εφαρμογών υιοθέτησε γρήγορα αρχές μηχανικής λογισμικού, όπως η σχεδίαση προσανατολισμένη στα συστατικά στοιχεία και τα τυπικά στοιχεία. Συγκεκριμένα, χαρακτηριστικά όπως η αναζήτηση, η διανομή και η προσθήκη ετικετών έχουν γίνει τυποποιημένα στοιχεία σε ένα νέο κύμα συνεργατικών εφαρμογών και

διαδικασιών. Η μελλοντική εξέλιξη των διαδικτυακών εφαρμογών θα κατευθύνεται από την πρόοδο στην τεχνολογία του προγράμματος περιήγησης, την υποδομή του Διαδικτύου, τα πρότυπα πρωτοκόλλων, τις πρακτικές μηχανικής λογισμικού και τις αναδυόμενες τάσεις εφαρμογών.

Μια εφαρμογή Ιστού ορίζεται ως μια εφαρμογή που βασίζεται σε HTTP με αλληλεπιδράσεις που μπορούν να υποβληθούν σε επεξεργασία από διακομιστές. Ενώ πολλοί υπάρχοντες ιστότοποι αποτελούν παραδείγματα δυναμικών εφαρμογών που βασίζονται σε HTTP, ένα σημαντικό μέρος τους βασίζεται στις ανθρώπινες γνωστικές ικανότητες για αποτελεσματική και ισχυρή χρήση. Τυπικές διαδικτυακές εφαρμογές:

- Βασίζονται στην υπάρχουσα αρχιτεκτονική και υποδομή ιστού.
- Δεν εξαρτώνται από συγκεκριμένες πλατφόρμες ή γλώσσες προγραμματισμού.
- Ενθαρρύνετε την επαναχρησιμοποίηση εφαρμογών πέρα από τα προγράμματα περιήγησης ιστού.
- Διευκολύνετε την ενσωμάτωση με άλλες εφαρμογές ιστού ή επιτραπέζιου υπολογιστή.
- Ζητήστε σημασιολογική σαφήνεια στο περιεχόμενο (παραστάσεις) που ανταλλάσσονται κατά τη χρήση τους.

Η ικανοποίηση της τελευταίας απαίτησης μπορεί να επιτευχθεί με τη χρήση XML, είτε μέσω του καθορισμού ενός ολοκληρωμένου προσαρμοσμένου σχήματος προσαρμοσμένου στον τομέα εφαρμογής είτε με την ενσωμάτωση μιας προσαρμοσμένης μικρο-μορφής σε ένα υπάρχον σχήμα χρησιμοποιώντας τα σημεία επεκτασιμότητας του.

Μια εφαρμογή Ιστού αποτελείται ουσιαστικά από καθαρό κώδικα HTML και CSS βελτιστοποιημένο για διάφορες συσκευές. Λειτουργεί ως ένας ιστότοπος με περιορισμένο περιεχόμενο και λειτουργικότητα, σχεδιασμένος να διευκολύνει την πλοήγηση των χρηστών κυρίως σε οθόνες αφής, χωρίς να βασίζεται σε περιφερειακά όπως το ποντίκι και το πληκτρολόγιο. Επιπλέον, εξασφαλίζει μια βελτιστοποιημένη εμπειρία ελαχιστοποιώντας τις λήψεις δεδομένων, όπως μεγάλες εικόνες και άλλα εκτεταμένα γραφικά στοιχεία..

Υβριδικές Διαδικτυακές Εφαρμογές - Hybrid Web Apps 2.3

Οι υβριδικές διαδικτυακές εφαρμογές (HWA) είναι ένας τύπος εφαρμογής λογισμικού που συνδυάζει στοιχεία τόσο εφαρμογών Ιστού όσο και εγγενών εφαρμογών. Οι HWA κατασκευάζονται με χρήση τεχνολογιών Ιστού όπως HTML, CSS και JavaScript, αλλά είναι συσκευασμένα σε ένα κέλυφος εγγενούς εφαρμογής. Αυτό τους επιτρέπει να εκτελούνται

εκτός σύνδεσης, να έχουν πρόσβαση στις λειτουργίες της εγγενούς συσκευής και να παρέχουν μια πιο απρόσκοπτη εμπειρία χρήστη από τις παραδοσιακές εφαρμογές Ιστού.

Οι υβριδικές διαδικτυακές εφαρμογές γίνονται όλο και πιο δημοφιλείς λόγω μιας σειράς πλεονεκτημάτων που προσφέρουν τόσο σε εφαρμογές ιστού όσο και σε εγγενείς εφαρμογές. Πρώτον, οι HWA είναι πιο οικονομικά στην ανάπτυξη από τις εγγενείς εφαρμογές, καθώς οι προγραμματιστές χρειάζεται μόνο να γράψουν μία βάση κώδικα για όλες τις πλατφόρμες-στόχους. Δεύτερον, είναι ευκολότερη η συντήρηση και η ενημέρωση τους σε σχέση με τις εφαρμογές web, καθώς οι προγραμματιστές χρειάζεται να ενημερώσουν τη βάση κώδικα μόνο μία φορά και οι αλλαγές θα αντικατοπτρίζονται σε όλες τις εκδόσεις της εφαρμογής. Τρίτον, οι HWA μπορούν να παρέχουν καλύτερη εμπειρία στον χρήστη από τις εφαρμογές Ιστού, καθώς μπορούν να εκτελούνται εκτός σύνδεσης, να έχουν πρόσβαση στις λειτουργίες της εγγενούς συσκευής και να παρέχουν μια πιο εγγενή εμφάνιση και αίσθηση[2].

Μερικά δημοφιλή παραδείγματα HWA περιλαμβάνουν το Twitter, το Instagram και το Gmail. Όλες αυτές οι εφαρμογές χρησιμοποιούν τεχνολογίες Ιστού για να δημιουργήσουν τις διεπαφές πελάτη και τη λειτουργικότητά τους, αλλά συμπεριλαμβάνονται σε εγγενείς εφαρμογές, έτσι ώστε να μπορούν να επωφεληθούν από τα οφέλη των εγγενών εφαρμογών.

Τα πλεονεκτήματα 2.3

- **Ανάπτυξη App Store:** Οι υβριδικές εφαρμογές μπορούν να δημοσιεύονται σε καταστήματα εφαρμογών, επιτρέποντας πωλήσεις, διαφημίσεις και αγορές εντός εφαρμογής.
- **Συμβατότητα μεταξύ πλατφορμών:** Οι υβριδικές εφαρμογές μπορούν να στοχεύουν πολλαπλές πλατφόρμες όπως iOS, Android και Windows.
- **Λειτουργίες συσκευής:** Οι υβριδικές εφαρμογές έχουν πρόσβαση σε εγγενή API συσκευών για λειτουργίες όπως κάμερα, γεωγραφική τοποθεσία και άλλα.
- **Δυνατότητα εκτός σύνδεσης:** Οι υβριδικές εφαρμογές μπορούν να λειτουργούν εκτός σύνδεσης, χάρη στην πρόσβασή τους σε API συσκευών, παρέχοντας μια εμπειρία που μοιάζει με εγγενή.
- **Βελτιωμένη απόδοση:** Οι υβριδικές εφαρμογές έχουν γενικά καλύτερη απόδοση από τις εφαρμογές ιστού.
- **Απόδοση κόστους και χρόνου:** Η ανάπτυξη υβριδικών εφαρμογών είναι οικονομική και χρονικά αποδοτική, ειδικά όταν χρησιμοποιούνται τυπικές τεχνολογίες ιστού όπως HTML, CSS και JavaScript. Πρόσθετες λειτουργίες ενδέχεται να αυξήσουν τον χρόνο ανάπτυξης αλλά και να βελτιώσουν την ποιότητα και την πολυπλοκότητα της εφαρμογής.

ΚΕΦΑΛΑΙΟ 3 Περιγραφή του περιβάλλοντος ανάπτυξης

Server-Side Programming (Backend) 3.1

MySQL 3.1.1

Οι βάσεις δεδομένων διαχειρίζονται από ένα σύστημα διαχείρισης βάσεων δεδομένων (DBMS) ή από έναν διακομιστή βάσης δεδομένων. Ένας διακομιστής βάσης δεδομένων διευκολύνει μια γλώσσα βάσης δεδομένων για τη δημιουργία, τη διαγραφή, τη διαχείριση και την ανάκτηση δεδομένων. Σχεδόν όλοι οι διακομιστές βάσεων δεδομένων χρησιμοποιούν SQL, η οποία αποτελείται από δηλώσεις για τον ορισμό και τον χειρισμό δεδομένων. Μόλις δημιουργηθεί μια βάση δεδομένων, οι πιο συχνά χρησιμοποιούμενες προτάσεις SQL περιλαμβάνουν INSERT, UPDATE, DELETE και SELECT, που χρησιμοποιούνται για την προσθήκη, τροποποίηση, αφαίρεση και αναζήτηση δεδομένων σε μια βάση δεδομένων, αντίστοιχα.

Αντί να αποθηκεύει όλα τα δεδομένα σε ένα ενιαίο χώρο αποθήκευσης, μια σχεσιακή βάση δεδομένων διαιρεί τα δεδομένα σε ξεχωριστούς πίνακες. Αυτές οι δομές βάσης δεδομένων είναι οργανωμένες σε βελτιστοποιημένα φυσικά αρχεία για τη βελτίωση της ταχύτητας. Το λογικό μοντέλο περιλαμβάνει αντικείμενα όπως βάσεις δεδομένων, πίνακες, προβολές, γραμμές και στήλες, παρέχοντας ένα ευέλικτο περιβάλλον προγραμματισμού. Επιτρέπει τη θέσπιση κανόνων που διέπουν τις σχέσεις μεταξύ πεδίων δεδομένων, συμπεριλαμβανομένων ενός προς ένα, ενός προς πολλά, μοναδικών, απαιτούμενων, προαιρετικών και δεικτών μεταξύ διαφορετικών πινάκων. Η βάση δεδομένων επιβάλλει αυτούς τους κανόνες, διασφαλίζοντας ότι μια καλά σχεδιασμένη βάση δεδομένων αποτρέπει τις εφαρμογές από το να αντιμετωπίσουν ασυνεπή, διπλά, ορφανά, παρωχημένα ή ελλιπή δεδομένα.

Το τμήμα "SQL" του "MySQL" σημαίνει "Structured Query Language", η οποία είναι η πιο ευρέως αναγνωρισμένη τυποποιημένη γλώσσα για πρόσβαση σε βάσεις δεδομένων. Ανάλογα με το περιβάλλον προγραμματισμού, είναι δυνατό είτε να εισάγουμε απευθείας SQL (π.χ. για τη δημιουργία αναφορών), να ενσωματώσουμε δηλώσεις SQL σε κώδικα γραμμένο σε άλλη γλώσσα ή να χρησιμοποιήσουμε ένα API συγκεκριμένης γλώσσας που κρύβει τη σύνταξη SQL.

Η MySQL χρησιμεύει ως διακομιστής σχεσιακής βάσης δεδομένων που υποστηρίζει την ευρέως χρησιμοποιούμενη γλώσσα SQL. Η MySQL συγκαταλέγεται στις πιο δημοφιλείς βάσεις δεδομένων στον κόσμο κυρίως λόγω:

- της φύσης ανοιχτού κώδικα
- της αξιοπιστίας
- της συμβατότητάς της με τους μεγάλους παρόχους φιλοξενίας
- Είναι μια οικονομικά αποδοτική βάση δεδομένων
- Είναι εύκολη στη διαχείριση.

Ωστόσο, η MySQL έχει ορισμένους περιορισμούς όσον αφορά την επεκτασιμότητα, όπως ο εκτεταμένος χρόνος ανάπτυξης και το κόστος καταγραφής της βάσης δεδομένων.

PHP 3.1.2

Η PHP, που σημαίνει "PHP: Hypertext Preprocessor" σε ένα αναδρομικό αρκτικόλεξο, είναι μια ευρέως αποδεκτή γλώσσα δέσμης εντολών ανοιχτού κώδικα με ένα ευρύ φάσμα εφαρμογών. Είναι ιδιαίτερα κατάλληλη για ανάπτυξη εφαρμογών ιστού και μπορεί να ενσωματωθεί απρόσκοπτα σε HTML[2].

Σε αντίθεση με γλώσσες όπως η C ή η Perl που βασίζονται σε πολλές εντολές για την παραγωγή εξόδου HTML, οι σελίδες PHP συνδυάζουν HTML με ενσωματωμένο κώδικα που εκτελεί συγκεκριμένες εργασίες. Σε αυτό το πλαίσιο, ο κώδικας εντός της PHP περικλείεται σε διακριτές οδηγίες επεξεργασίας έναρξης και λήξης, `<?php` και `?>`, επιτρέποντας τη μετάβαση μέσα και έξω από τη "λειτουργία PHP"[7].

```
<!DOCTYPE html>
<html>
<body>

<h1>Developer News</h1>

<?php echo "The Best PHP Examples"; ?>

</body>
</html>
```

Εικόνα 1. Παράδειγμα κώδικα PHP ενσωματωμένο σε κώδικα HTML.

Αυτό που κάνει την PHP να ξεχωρίζει από κάτι σαν την JavaScript είναι ότι ο κώδικας PHP εκτελείται στον διακομιστή, παράγοντας περιεχόμενο HTML που αποστέλλεται στον χρήστη. Ως αποτέλεσμα, ο χρήστης λαμβάνει την έξοδο της δέσμης εντολών, αλλά δεν γνωρίζει τον υποκείμενο κώδικα. Επιπλέον, έχετε την επιλογή να διαμορφώσετε τον διακομιστή ιστού σας ώστε να χειρίζεται όλα τα αρχεία HTML χρησιμοποιώντας PHP, αποκρύπτοντας αποτελεσματικά τη φύση των λειτουργιών της εφαρμογής ιστού από τους χρήστες.

Laravel PHP Framework 3.1.3

Το Laravel είναι ένα δωρεάν, ανοιχτού κώδικα πλαίσιο ανάπτυξης ιστοσελίδων σε PHP σχεδιασμένο για την κατασκευή διαδικτυακών εφαρμογών με βάση το αρχιτεκτονικό πρότυπο Model-View-Controller (MVC)[8].

Το πλαίσιο προσφέρει ένα σύστημα συσκευασίας με εξειδικευμένο διαχειριστή εξαρτήσεων, ενώ παρέχει ενσωματωμένα εργαλεία για πιστοποίηση, δρομολόγηση, διαχείριση συνεδριών, κρυφή μνήμη, δοχείο IoC, ενώ παρέχει και εργαλεία διαχείρισης βάσεων δεδομένων και υποστήριξη για ενσωματωμένα δοκιμαστικά[3].

Με έμφαση στην απλότητα και ευελιξία, το Laravel έχει αποκτήσει το μεγαλύτερο μερίδιο στην αγορά των PHP MVC πλαισίων. Αναδεικνύεται ως πλήρες πλαίσιο ανάπτυξης, καθώς μπορεί να έχει τη διαχείριση από τον διακομιστή ιστού έως τη διαχείριση βάσεων δεδομένων και τη δημιουργία HTML.

Το Laravel συνδυάζει ασφάλεια και ευελιξία, προσφέροντας προηγμένες δυνατότητες, όπως προσαρμοσμένες διαδρομές HTTPS, ενώ εξασφαλίζει την προστασία των δεδομένων με αυτόματη προσθήκη https:// πριν από το URL. Επιπλέον, ο Query Builder του Laravel παρέχει ένα εύελικτο περιβάλλον για τη δημιουργία και εκτέλεση ερωτημάτων βάσης δεδομένων, προστατεύοντας από επιθέσεις SQL injection.

Το Laravel αποτελεί μια ισχυρή επιλογή για την ανάπτυξη δυναμικών ιστοσελίδων, προσφέροντας ένα ολοκληρωμένο περιβάλλον ανάπτυξης με συνεχείς βελτιώσεις και υποστήριξη από την κοινότητα ανοιχτού κώδικα

JavaScript Object Notation 3.1.4

Το JavaScript Object Notation (JSON) είναι μια μορφή κειμένου που χρησιμοποιείται για τη σειριοποίηση δομημένων δεδομένων. Το JSON μπορεί να αντιπροσωπεύει τέσσερις θεμελιώδεις τύπους, οι οποίοι περιλαμβάνουν:

- συμβολοσειρές,
- αριθμούς,
- boolean
- null
- μαζί με δύο δομημένους τύπους:
 - ο αντικείμενα
 - ο πίνακες.

Μια συμβολοσειρά ορίζεται ως μια διαδοχή μηδέν ή περισσότερων χαρακτήρων Unicode [UNICODE], αξίζει να σημειωθεί ότι αυτή η παραπομπή αναφέρεται στην πιο πρόσφατη επανάληψη της προδιαγραφής Unicode χωρίς να προσδιορίζει μια συγκεκριμένη έκδοση. Η σύνταξη του JSON δεν αναμένεται να επηρεαστεί από μελλοντικές αλλαγές στην προδιαγραφή Unicode[12].

```
[
  {
    "precision": "zip",
    "Latitude": 37.7668,
    "Longitude": -122.3959,
    "Address": "",
    "City": "SAN FRANCISCO",
    "State": "CA",
    "Zip": "94107",
    "Country": "US"
  },
  {
    "precision": "zip",
    "Latitude": 37.371991,
    "Longitude": -122.026020,
    "Address": "",
    "City": "SUNNYVALE",
    "State": "CA",
    "Zip": "94085",
    "Country": "US"
  }
]
```

Εικόνα 2. παράδειγμα πίνακα JSON με 2 αντικείμενα

Ένα αντικείμενο εντός JSON είναι μια μη ταξινομημένη συλλογή μηδενικών ή περισσότερων ζευγών ονόματος/τιμής, όπου ένα όνομα είναι μια συμβολοσειρά και μια τιμή θα μπορούσε να είναι μια συμβολοσειρά, αριθμός, boolean, null, αντικείμενο ή πίνακας. Το κείμενο JSON είναι ουσιαστικά μια ακολουθία από διακριτικά, που περιλαμβάνει έξι δομικούς χαρακτήρες, συμβολοσειρές, αριθμούς και τρία κυριολεκτικά ονόματα. Στην ουσία, ένα κείμενο JSON αντιπροσωπεύει μια σειριακή τιμή.

[API 3.1.5](#)

Ένα Application Programming Interface (API) χρησιμεύει ως ενδιάμεσο λογισμικό που διευκολύνει την επικοινωνία μεταξύ δύο εφαρμογών. Επιτρέπει επίσης την αλληλεπίδραση και τον συγχρονισμό δεδομένων μεταξύ ενός προϊόντος ή μιας υπηρεσίας και μιας άλλης εφαρμογής. Ως αποτέλεσμα, υπάρχουν πολλά πλεονεκτήματα για τους προγραμματιστές όταν επιλέγουν να χρησιμοποιούν ένα υπάρχον API αντί να δημιουργούν μια εντελώς νέα λύση λογισμικού κάθε φορά που απαιτείται μια νέα δυνατότητα. Αυτή η απόφαση μπορεί

να αποτελέσει δίλημμα για τους προγραμματιστές. Επιπλέον, είναι σημαντικό να σημειωθεί ότι ο πρωταρχικός στόχος της εφαρμογής ενός νέου API είναι η απλούστευση της διαδικασίας ανάπτυξης και η ελαχιστοποίηση των επαναλαμβανόμενων εργασιών που εμπλέκονται στη μετάδοση δεδομένων μεταξύ διαφορετικών στοιχείων υπολογιστή[6].

Representational State Transfer (Rest API) 3.1.6

Το REST (Representational State Transfer) είναι μια αρχιτεκτονική διεπαφή προγραμματισμού εφαρμογών (API) που διευκολύνει την επικοινωνία πελάτη-διακομιστή για εφαρμογές web μέσω του πρωτοκόλλου HTTP. Προσφέρει ευελιξία καθώς δεν συνδέεται με κάποιο συγκεκριμένο πρωτόκολλο μεταφοράς. Το REST καθοδηγείται από τρεις βασικές αρχές σχεδιασμού: διευθυνσιοδότηση, ομοιόμορφη διεπαφή και ανιθαγένεια. Η διαχείριση της διευθυνσιοδότησης γίνεται με τη δημιουργία τελικών σημείων σε έναν δομημένο κατάλογο χρησιμοποιώντας διάφορα URI για την ανάκτηση δεδομένων. Το API λειτουργεί βάσει του παραδείγματος CRUD (Δημιουργία, Ανάγνωση, Ενημέρωση, Διαγραφή), το οποίο ευθυγραμμίζεται με κοινές λειτουργίες, συμπεριλαμβανομένων των INSERT, SELECT, UPDATE και DELETE, που χρησιμοποιούνται συνήθως σε μόνιμα συστήματα αποθήκευσης δεδομένων όπως το SQL.

Η πρόσβαση σε μια υπηρεσία RESTful μέσω του πρωτοκόλλου HTTP είναι δυνατή μέσω διαφόρων γλωσσών προγραμματισμού. Για παράδειγμα, στο jQuery, ένα πλαίσιο JavaScript, ένας διακομιστής REST μπορεί να κληθεί χρησιμοποιώντας ένα ερώτημα Ajax. Η τυπική μορφή απόκρισης δεδομένων είναι JSON (JavaScript Object Notation).

Axios Api 3.1.7

Το Axios είναι ένα πρόγραμμα-πελάτη HTTP που βασίζεται σε JavaScript που λειτουργεί με υποσχέσεις και βρίσκει χρησιμότητα τόσο στο front-end όσο και στο back-end τμήμα μιας εφαρμογής. Απλοποιεί τις αλληλεπιδράσεις με τα RESTful API και διευκολύνει ενέργειες όπως η δημιουργία, η ανάγνωση, η ενημέρωση και η διαγραφή δεδομένων. Το Axios είναι συμβατό με την υποβολή αιτημάτων HTTP σε περιβάλλον node.js. Επιπλέον, προσφέρει δυνατότητες για υποκλοπή και τροποποίηση αιτημάτων και απαντήσεων. Συγκεκριμένα, το Axios αυτοματοποιεί επίσης τη μετατροπή δεδομένων σε JSON (JavaScript Object Notation)[15].

Client-Side Programming (Frontend) 3.2

HTML 3.2.1

Η HTML, ή HyperText Markup Language, χρησιμεύει ως το θεμελιώδες θεμέλιο του Διαδικτύου. Καθορίζει τον σκοπό και τη διάταξη του διαδικτυακού περιεχομένου. Ενώ η HTML ασχολείται με περιεχόμενο, πρόσθετες τεχνολογίες όπως το CSS χειρίζονται τον οπτικό σχεδιασμό ιστοσελίδων και η JavaScript διαχειρίζεται τη λειτουργικότητα και τη συμπεριφορά τους[16].

Ο όρος "HyperText" υποδηλώνει τους διασυνδεδεμένους συνδέσμους που συνδέουν ιστοσελίδες μεταξύ τους, είτε μέσα σε έναν ενιαίο ιστότοπο είτε σε διαφορετικούς ιστότοπους. Οι σύνδεσμοι είναι ένα κρίσιμο συστατικό του Διαδικτύου, που αποτελούν τη βασική δομή του. Όταν ανεβάζετε περιεχόμενο στον Ιστό και το συνδέετε με σελίδες που έχουν δημιουργηθεί από άλλους, συμμετέχετε ενεργά στον Παγκόσμιο Ιστό.

JavaScript 3.2.2

Η JavaScript, που συνήθως αναφέρεται ως "JS", είναι μια ευέλικτη γλώσσα δέσμης εντολών ή προγραμματισμού. Επιτρέπει την εφαρμογή περίπλοκων λειτουργιών σε ιστοσελίδες. Κάθε φορά που μια ιστοσελίδα υπερβαίνει την παρουσίαση στατικού περιεχομένου, συμμετέχοντας σε δραστηριότητες όπως ενημερώσεις σε πραγματικό χρόνο, διαδραστικούς χάρτες, δυναμικά γραφικά 2D/3D ή ροή βίντεο, η JavaScript παίζει συνήθως καθοριστικό ρόλο. Αποτελεί το τρίτο επίπεδο στην ιεραρχία των τυπικών τεχνολογιών Ιστού, με τα HTML και CSS να είναι τα δύο προηγούμενα επίπεδα, τα οποία έχουμε διερευνήσει εκτενέστερα σε προηγούμενες ενότητες της Περιοχής Μάθησης[13].

Η JavaScript χρησιμοποιείται κυρίως για δυναμική δέσμη εντολών από την πλευρά του πελάτη εντός ιστοσελίδων, αλλά χρησιμοποιείται συχνά και από την πλευρά του διακομιστή, διευκολύνεται από χρόνους εκτέλεσης όπως το Node.js.

Cascading Style Sheets (CSS) 3.2.3

Τα Cascading Style Sheets (CSS) είναι μια δηλωτική γλώσσα που ελέγχει την οπτική παρουσίαση ιστοσελίδων σε προγράμματα περιήγησης ιστού. Η CSS είναι υπεύθυνη για την εφαρμογή κανόνων στυλ σε καθορισμένα στοιχεία για να διασφαλιστεί η σωστή απόδοσή τους. Κάθε κανόνας στυλ περιλαμβάνει ιδιότητες και τις αντίστοιχες τιμές τους, οι οποίες

υπαγορεύουν την εμφάνιση μιας ιστοσελίδας. Το CSS αποτελεί μια από τις θεμελιώδεις τεχνολογίες για τον Παγκόσμιο Ιστό, μαζί με την HTML και τη JavaScript.

Κατά κύριο λόγο, το CSS προσφέρει τη δυνατότητα δημιουργίας πολύ πιο διαφορετικών και εξελιγμένων οπτικών σχεδίων για έγγραφα όπως το HTML. Με το CSS, μπορείτε να καθορίσετε χρώματα κειμένου και φόντου για οποιοδήποτε στοιχείο, να ορίσετε περιγράμματα γύρω από στοιχεία, να προσαρμόσετε τα κενά γύρω από τα στοιχεία, να ελέγξετε τη χρήση κεφαλαίων κειμένου, τη διακόσμηση κειμένου (π.χ. υπογράμμιση), την απόσταση γραμμών και ακόμη και να καθορίσετε εάν το κείμενο εμφανίζεται ή κρύβεται. Επιτρέπει επίσης την υλοποίηση πολλών άλλων οπτικών εφέ[17].

Vue.js Πλαίσιο της JavaScript 3.2.4

Το Vue είναι ένα πλαίσιο JavaScript που χρησιμοποιείται για τη δημιουργία διεπαφών χρήστη. Είναι χτισμένο στη βάση του τυπικού HTML, CSS και JavaScript, προσφέροντας ένα δηλωτικό και βασισμένο σε στοιχεία μοντέλο προγραμματισμού για να διευκολύνει την αποτελεσματική ανάπτυξη διεπαφών χρήστη, είτε είναι απλές είτε περίπλοκες.

Οι δύο θεμελιώδεις πτυχές του Vue[9]:

- **Δηλωτική απόδοση:** Το Vue επεκτείνεται σε τυπικό HTML εισάγοντας μια σύνταξη προτύπου που μας επιτρέπει να εκφράσουμε την έξοδο HTML δηλωτικά με βάση την κατάσταση JavaScript.
- **Αντιδραστικότητα:** Το Vue παρακολουθεί αυτόματα τις αλλαγές στην κατάσταση της JavaScript και ενημερώνει αποτελεσματικά το Μοντέλο Αντικειμένου Εγγράφου (DOM) όταν συμβαίνουν τέτοιες αλλαγές.

Το Vue είναι ένα ολοκληρωμένο πλαίσιο και οικοσύστημα που ανταποκρίνεται στις περισσότερες από τις τυπικές απαιτήσεις στην ανάπτυξη του frontend. Ωστόσο, ο Ιστός παρουσιάζει μεγάλη ποικιλία και η φύση και η κλίμακα των εφαρμογών Ιστού μπορεί να διαφέρουν σημαντικά. Το Vue έχει σχεδιαστεί έχοντας αυτό κατά νου, δίνοντας έμφαση στην ευελιξία και μια σταδιακή προσέγγιση υιοθέτησης. Ανάλογα με τη συγκεκριμένη περίπτωση χρήσης, μπορείτε να χρησιμοποιήσετε το Vue με διάφορους τρόπους:

- Βελτίωση στατικού HTML χωρίς να απαιτείται διαδικασία κατασκευής.
- Ενσωμάτωσή του ως Στοιχεία Ιστού σε οποιαδήποτε ιστοσελίδα.
- Ανάπτυξη Εφαρμογών Μονής Σελίδας (SPAs).
- Χρησιμοποιώντας το για Fullstack και Rendering από την πλευρά του διακομιστή (SSR).

- Στόχευση διαφορετικών πλατφορμών, συμπεριλαμβανομένων των επιτραπέζιων υπολογιστών, των κινητών, του WebGL, ακόμη και του τερματικού.

Παρά την προσαρμοστικότητά του, το Vue μοιράζεται ένα βασικό σύνολο αρχών σε όλα αυτά τα σενάρια εφαρμογών. Το Vue αναφέρεται συχνά ως "Το προοδευτικό πλαίσιο" - είναι ένα πλαίσιο που μπορεί να εξελιχθεί και να προσαρμοστεί παράλληλα με τις διαρκώς μεταβαλλόμενες απαιτήσεις σας.

Pinia 3.2.5

Το Pinia ξεκίνησε ως ένα πειραματικό έργο, με στόχο να επαναπροσδιορίσει την έννοια του Store για το Vue, υιοθετώντας το Composition API. Το API παραμένει συνεπές και για τις δύο εκδόσεις, με τις διακρίσεις να περιορίζονται κυρίως στην εγκατάσταση και την απόδοση από την πλευρά του διακομιστή. Ουσιαστικά, το Pinia λειτουργεί ως βιβλιοθήκη καταστήματος για το Vue, επιτρέποντάς σας να μοιράζεστε κατάσταση μεταξύ στοιχείων και σελίδων[18].

Quasar Πλαίσιο της JavaScript 3.2.6

Το Quasar είναι ένα πλαίσιο ανοιχτού κώδικα που βασίζεται στο Vue.js. Αυτό το πλαίσιο δίνει τη δυνατότητα στους προγραμματιστές ιστού να δημιουργούν γρήγορα ευέλικτους ιστότοπους και εφαρμογές σε διάφορες μορφές, όπως[11]:

- SPA (Εφαρμογές μιας σελίδας)
- SSR (Server-side Rendered Applications) με προαιρετική εξαγορά πελάτη PWA
- PWA (προοδευτικές εφαρμογές Ιστού)
- Εφαρμογές για κινητά για Android, iOS και άλλα, που μπορούν να επιτευχθούν μέσω Cordova ή Capacitor
- Εφαρμογές επιφάνειας εργασίας πολλαπλών πλατφορμών με χρήση Electron.js
- Είναι χτισμένο πάνω στο Vue.js
- Προσφέρει εξαιρετική υποστήριξη τόσο για προγράμματα περιήγησης για υπολογιστές όσο και για κινητά, συμπεριλαμβανομένου του iOS Safari, χωρίς να απαιτείται επιπλέον διαμόρφωση.
- Το Quasar είναι εξαιρετικά προσαρμόσιμο μέσω CSS και επεκτάσιμο μέσω JavaScript, επιτρέποντας στους προγραμματιστές να το προσαρμόσουν στις ανάγκες τους.
- Δίνει προτεραιότητα στην απόδοση, καθιστώντας το ένα από τα πιο εστιασμένα, στην απόδοση, διαθέσιμα πλαίσια.

Σε κάθε κατασκευή μιας εφαρμογής, το Quasar θα δημιουργήσει μόνο τον απαραίτητο κώδικα για τον συγκεκριμένο τύπο εφαρμογής, εξαιρώντας τυχόν περιττά στοιχεία.

Electron.js 3.2.7

Το Electron είναι ένα πλαίσιο ανοιχτού κώδικα σχεδιασμένο για τη δημιουργία επιτραπέζιων εφαρμογών χρησιμοποιώντας γλώσσες ανάπτυξης ιστού. Το Electron JS αξιοποιεί έναν συνδυασμό δύο βασικών τεχνολογιών: του χρόνου εκτέλεσης Node.js και του προγράμματος περιήγησης ιστού Chromium.

Όταν εκκινείται μια εφαρμογή Electron, λειτουργεί με δύο διακριτές διαδικασίες: την κύρια διαδικασία και τη διαδικασία απόδοσης εικόνας. Η κύρια διαδικασία φροντίζει για τη λειτουργικότητα του backend της εφαρμογής Electron και έχει εγγενώς πρόσβαση στα Node.js API. Σε κάθε περίπτωση, υπάρχει μόνο μία κύρια διαδικασία που είναι υπεύθυνη για τη δημιουργία στοιχείων γραφικής διεπαφής χρήστη και την επίβλεψη των διαδικασιών απόδοσης εικόνας[19].

ΚΕΦΑΛΑΙΟ 4 Περιγραφή της εφαρμογής σχεδιασμός και ανάπτυξη της βάσης δεδομένων

Εισαγωγή 4.1

Σε αυτό το κεφάλαιο θα γίνει εκτενής ανάλυση όσον αφορά τον σχεδιασμό της εφαρμογής και της βάσης δεδομένων που χρησιμοποιήσα.

Σχεδίαση βάσης δεδομένων 4.2

Η ανάγκες που προέκυψαν για αποθήκευση δεδομένων στην εφαρμογή οδήγησαν στη δημιουργία μίας βάσης δεδομένων. Αυτή αποτελείται από τρεις πίνακες, τον πίνακα `users` όπου αποθηκεύονται όλα τα στοιχεία των χρηστών, τον πίνακα `reservations` όπου αποθηκεύονται όλες τις κρατήσεις των θέσεων στάθμευσης και τον πίνακα `parking_spots` αποθηκεύονται όλες οι θέσεις στάθμευσης που ανεβάζουν οι χρήστες. Συμπληρωματικά κατά τη δημιουργία της εφαρμογής, δημιουργήθηκαν αυτόματα οι εξής πίνακες:

- **failed_jobs:** η Laravel προσφέρει μια απλή μέθοδο για τον καθορισμό του μέγιστου αριθμού φορών που μπορεί να επιχειρηθεί μια διεργασία. Όταν μια διεργασία ξεπεράσει αυτό το καθορισμένο όριο προσπαθειών, θα καταγραφεί στον πίνακα της βάσης δεδομένων `failed_jobs`.
- **migrations:** αποθηκεύει τις τελευταίες εκδόσεις των υπολοίπων πινάκων που ονομάζονται `migrations`, έτσι είναι πολύ πιο εύκολο όταν μια ομάδα δουλεύει σε μια εφαρμογή να δουλεύουν όλοι με την ίδια βάση δεδομένων.
- **password_resets:** όταν ένας χρήστης έχει ξεχάσει τον κωδικό του και πρέπει να τον αλλάξει, αποστέλλεται ένα email στον χρήστη με ένα token, που είναι μοναδικό, έτσι ώστε να επαληθεύσουμε την ταυτότητά του. Αυτός ο πίνακας κρατάει το token που στείλαμε στο χρήστη και το email του χρήστη.
- **personal_access_tokens:** ένας πίνακας που αποθηκεύει το μοναδικό token του χρήστη που δημιουργείται κατά τη σύνδεση του στην εφαρμογή. Αυτό το token χρησιμοποιείται για να επαληθεύσουμε ποιος χρήστης συνδεδεμένος και για να κρατήσουμε το χρήστη συνδεδεμένο χωρίς να χρειάζεται να συνδέεται κάθε φορά που ανοίγει την εφαρμογή.



Εικόνα 3. Διάγραμμα Οντοτήτων-Συναρχειώσεων της Βάσης Δεδομένων

Βήματα δημιουργίας βάσης και πινάκων με την Laravel 4.2.1

Στο πρώτο βήμα χρησιμοποιούμε στο τερματικό την εντολή που φαίνεται στην παρακάτω εικόνα, σε αυτή την περίπτωση για να δημιουργήσουμε τον πίνακα που αποθηκεύει τους χρήστες, ότι γράφουμε μετά το create είναι το όνομα του πίνακα.

```
> php artisan make:migrations create_users_table
```

Εικόνα 4. Δήλωση νέου πίνακα με Laravel

Αποτέλεσμα αυτής της εντολής είναι να δημιουργηθεί το αρχείο φαίνεται στην παρακάτω εικόνα, στο οποίο δηλώνουμε όλες τις στήλες και το είδος μεταβλητής κάθε στήλης.

```

public function up()
{
    Schema::create('users', function (Blueprint $table) {
        $table->id();
        $table->string('name');
        $table->string('last_name');
        $table->string('username')->unique();
        $table->string('email')->unique();
        $table->integer('credits');
        $table->timestamp('email_verified_at')->nullable();
        $table->string('password');
        $table->rememberToken();
        $table->timestamps();
    });
}

/**
 * Reverse the migrations.
 *
 * @return void
 */
public function down()
{
    Schema::dropIfExists('users');
}

```

```
> php artisan migrate
```

Εικόνα 5. Εντολή για την δημιουργία του πίνακα στη βάση δεδομένων.

Μόλις έχουμε όλες οι μεταβλητές που θέλουμε και είμαστε έτοιμοι για να δημιουργήσουμε τον πίνακα εκτελούμε στο τερματικό την παρακάτω εντολή και δημιουργείται ο πίνακας στη βάση δεδομένων.

Ο πίνακας users 4.2.2

Στον πίνακα users αποθηκεύουμε ότι αφορά το χρήστη, οπότε προκύπτουν οι παρακάτω στήλες:

- **το id:** είναι τύπου `bigint(20)`, είναι το πρωτεύον κλειδί του πίνακα και δημιουργείται αυτόματα κατά την εγγραφή του χρήστη επίσης έχει όριομα `auto_increment` που το κάνει να αυξάνεται κατά ένα σε κάθε νέα εγγραφή χρήστη.
- **name:** είναι τύπου `varchar(255)` και αποθηκεύει το πραγματικό όνομα του χρήστη
- **last_name:** είναι τύπου `varchar(255)` αποθηκεύει το επώνυμο του χρήστη
- **username:** είναι τύπου `varchar(255)` αποθηκεύει το όνομα χρήστη
- **email:** είναι τύπου `varchar(255)` το email του χρήστη.
- **credits:** είναι τύπου `int(11)` τους διαθέσιμους πόντους του χρήστη.
- **password:** είναι τύπου `varchar(255)` τον κωδικό του χρήστη σε κρυπτογραφημένη μορφή, η κρυπτογράφηση γίνεται από την Laravel με την βιβλιοθήκη `bcrypt`.

Τέλος το `remember_token` που είναι τύπου `varchar(255)` και το `email_verified_at`, το `created_at` και το `updated_at` που είναι όλες μεταβλητές τύπου `timestamp` και δημιουργούνται αυτόματα από τη Laravel για την καλύτερη διαχείριση των δεδομένων του χρήστη.

Ο πίνακας parking_spots 4.2.3

Στον πίνακα `parking_spots` αποθηκεύουμε ότι αφορά τις θέσεις στάθμευσης οπότε προκύπτουν και οι παρακάτω στήλες:

- **το id:** είναι τύπου `bigint(20)`, είναι το πρωτεύον κλειδί του πίνακα και δημιουργείται αυτόματα κατά την εισαγωγή μιας νέας θέσης στάθμευσης, επίσης έχει όριομα `auto_increment` που το κάνει να αυξάνεται κατά ένα σε κάθε νέα εισαγωγή.
- **user_id:** είναι τύπου `bigint(20)` και δευτερεύον κλειδί του πίνακα. Αυτό το κλειδί χρησιμοποιείται στην ταύτιση της θέσης στάθμευσης με τον χρήστη που είναι ο ιδιοκτήτης της.
- **city:** είναι τύπου `varchar(255)` και αποθηκεύει την πόλη που βρίσκεται η θέση στάθμευσης

- **address:** είναι τύπου `varchar(255)` και αποθηκεύει τη διεύθυνση στις θέσεις στάθμευσης
- **title:** είναι τύπου `varchar` είναι τύπου(255) και αποθηκεύει τον τίτλο που ορίζει ο χρήστης στην θέση στάθμευσης
- **description:** είναι τύπου `varchar(255)` και αποθηκεύει την περιγραφή που ορίζει ο χρήστης για τη θέση στάθμευσης
- **vehicle_type:** είναι τύπου `enum('motorbike', 'car', 'suv', 'truck')` και διακρίνει την κατηγορία του οχήματος που χωράει στην θέση στάθμευσης

Ο πίνακας `reservations` 4.2.4

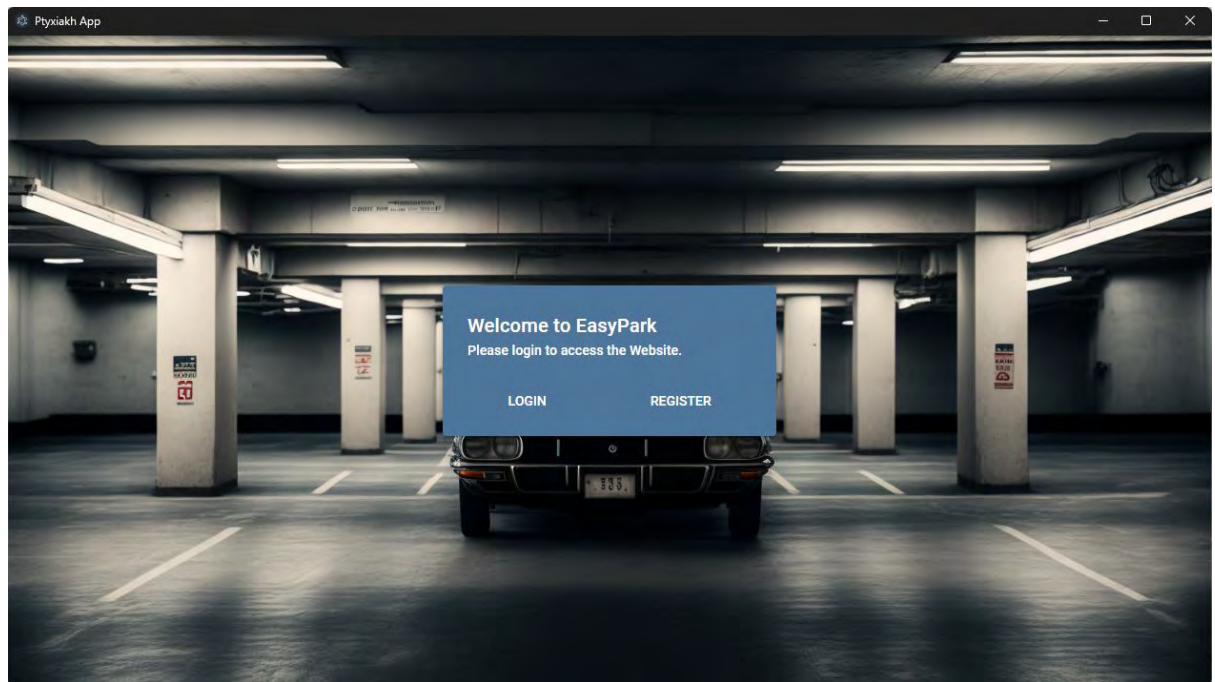
Στον πίνακα `reservations` αποθηκεύουμε όλες τις κρατήσεις που δημιουργούνται οπότε και προκύπτουν οι παρακάτω στήλες:

- **id:** είναι τύπου `bigint(20)`, είναι το πρωτεύον κλειδί του πίνακα και δημιουργείται αυτόματα κατά την εισαγωγή μιας νέας κράτησης θέσης στάθμευσης, επίσης έχει όρισμα `auto_increment` που το κάνει να αυξάνεται κατά ένα σε κάθε νέα εισαγωγή.
- **user_id:** είναι τύπου `bigint(20)` και δευτερεύον κλειδί του πίνακα. Αυτό το κλειδί χρησιμοποιείται στην ταύτιση της κράτησης με τον χρήστη που την πραγματοποίησε.
- **parking_spot_id:** είναι τύπου `bigint(20)` και δευτερεύον κλειδί του πίνακα. Αυτό το κλειδί χρησιμοποιείται στην ταύτιση της κράτησης με την θέση στάθμευσης στην οποία πραγματοποιήθηκε η κράτηση.
- **start_date:** είναι τύπου `datetime` και αποθηκεύει την ημερομηνία που ξεκινάει η κράτηση.
end_date: είναι τύπου `datetime` και αποθηκεύει την ημερομηνία που τελειώνει η κράτηση.

Περιγραφή της εφαρμογής 4.3

Landing page

Στην Εικόνα 6 παρουσιάζεται το landing page. Σε αυτή σελίδα ο χρήστης αντικρίζει μία κάρτα με δύο κουμπιά το `login` και το `register`. Ο χρήστης έχει τη δυνατότητα να πατήσει το `login` εάν διαθέτει ήδη λογαριασμό για την εφαρμογή και το `register` σε περίπτωση που δεν διαθέτει λογαριασμό για να δημιουργήσει ένα έτσι ώστε να μπορεί να εισέλθει στην εφαρμογή.



Εικόνα 6. Landing page

Login page

Στην Εικόνα 7 παρουσιάζεται το login page. Σε αυτή τη σελίδα απαιτείται από το χρήστη να συμπληρώσει το email και τον κωδικό του και να πατήσει το κουμπί login για να συνδεθεί και να εισέλθει στην εφαρμογή.

Login To Access Website

Email

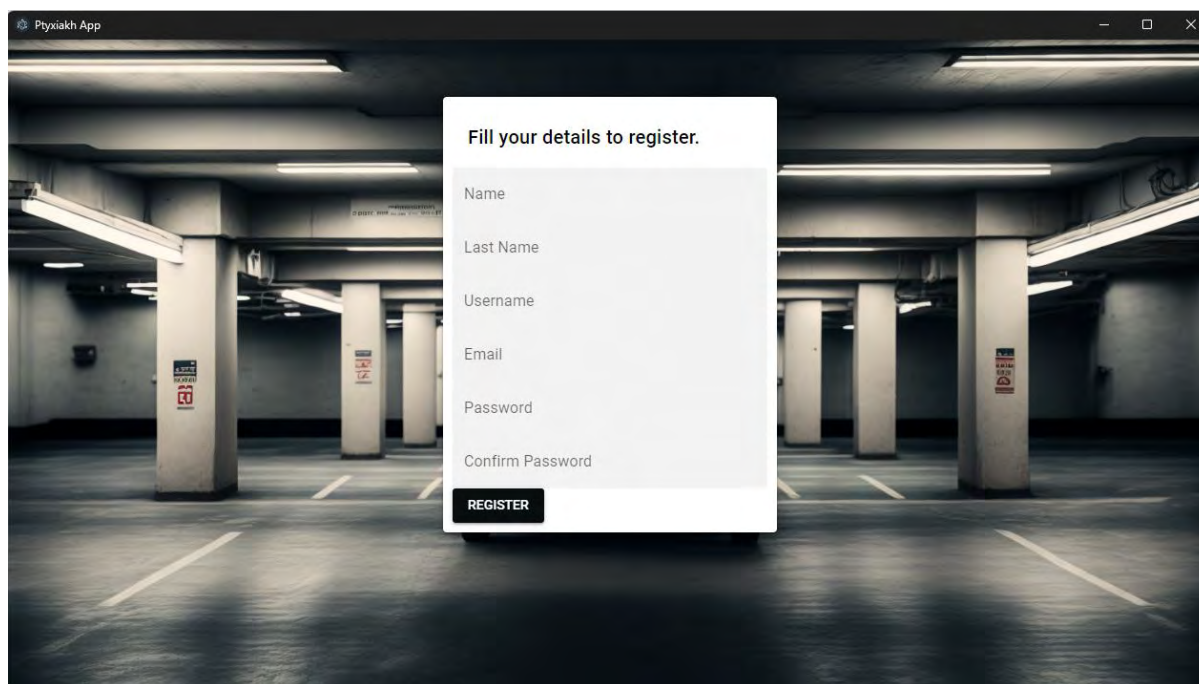
Password

LOGIN

Εικόνα 7. Login Page

Register Page

Στην Εικόνα 8 παρουσιάζεται το register page. Σε αυτή τη σελίδα ο χρήστης έχει την δυνατότητα να συμπληρώσει το όνομά του, το επίθετο του, ένα username, το email του και τον κωδικό που επιθυμεί. Επίσης το ζητείται να επιβεβαιώσει τον κωδικό του και τέλος μπορεί να πατήσει το κουμπί REGISTER για να δημιουργήσει ένα νέο λογαριασμό. Αν η δημιουργία του λογαριασμού είναι επιτυχής θα κατευθυνθεί στην αρχική σελίδα αυτόματα.



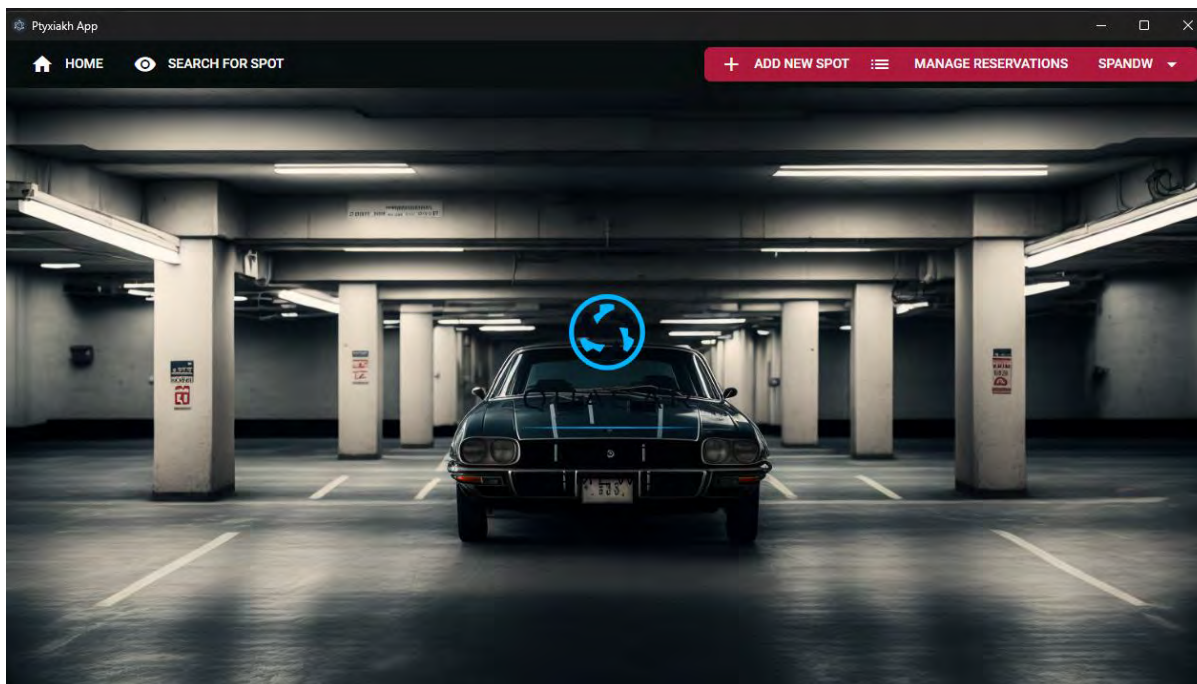
Εικόνα 8. Register Page

Home Page

Στην Εικόνα 9 παρουσιάζεται η αρχική σελίδα. Στην αρχική σελίδα της εφαρμογής στο πάνω μέρος της έχει ένα μενού με κουμπιά:

- Home: Το κουμπί που ανακατευθύνει τον χρήστη στο home page όπου και αν βρίσκεται στη σελίδα
- Για αναζήτηση θέσης(search for spot): αυτό το κουμπί μας οδηγεί στη λίστα με τις διαθέσιμες θέσεις στάθμευσης
- Προσθήκη Νέας θέσης: αυτό το κουμπί μας οδηγεί σε μία φόρμα για να προσθέσουμε μία νέα θέση στάθμευσης στην εφαρμογή
- Διαχείριση κρατήσεων: αυτό το κουμπί μας οδηγεί σε μία λίστα με τις ενεργές κρατήσεις που έχει πραγματοποιήσει ο χρήστης

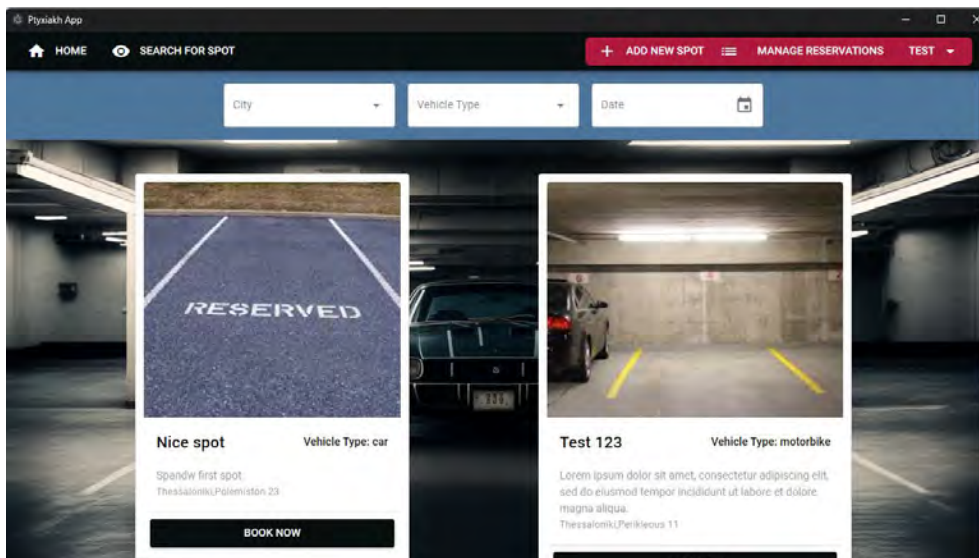
- Διαχείριση του λογαριασμού του χρήστη: Είναι ένα κουμπί με το username του χρήστη και μας οδηγεί στη σελίδα με το προφίλ του χρήστη



Εικόνα 9. Home page

Search for Spot

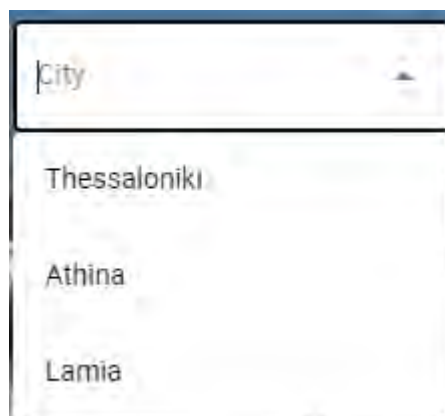
Στην Εικόνα 10 παρουσιάζεται η αρχική σελίδα. Σε αυτή σελίδα ο χρήστης έχει την δυνατότητα να αναζητά διαθέσιμες θέσεις στάθμευσης με την βοήθεια των τριών φίλτρων που παρουσιάζονται στην μπλε γραμμή.



Εικόνα 10. Σελίδα με τις διαθέσιμες θέσεις στάθμευσης.

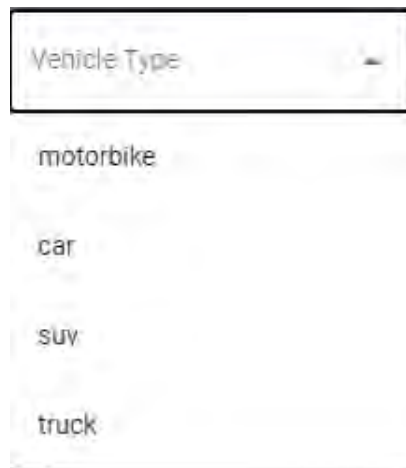
Ο χρήστης έχει την επιλογή να χρησιμοποιήσει τα εξής 3 φίλτρα αναζήτησης:

- Επιλογή της πόλης: Ο χρήστης μπορεί να επιλέξει από τις πόλεις που υπάρχουν διαθέσιμες θέσεις στάθμευσης όπως φαίνεται στην Εικόνα 11



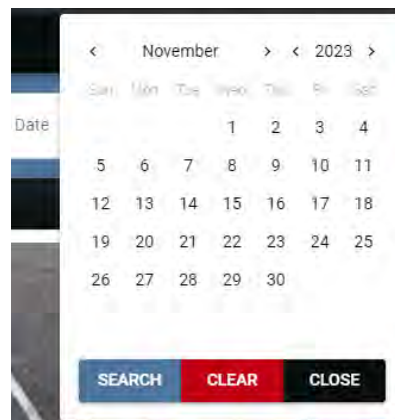
Εικόνα 11. Φίλτρο για την επιλογή αναζήτησης πόλης.

- Επιλογή του τύπου οχήματος: Ο χρήστης μπορεί να αναζητήσει μέσα από τέσσερις κατηγορίες οχημάτων ανάλογα με το όχημα που διαθέτει όπως φαίνεται στην Εικόνα 12



Εικόνα 12. Φίλτρο για την επιλογή αναζήτησης ανάλογα με τον τύπο του οχήματος.

- **Επιλογή των ημερομηνιών:** Ο χρήστης μπορεί να αναζητήσει σύμφωνα με τις ημερομηνίες που θέλει να βρει θέση στάθμευσης όπως φαίνεται στην Εικόνα 13



Εικόνα 13. Φίλτρο για την επιλογή ημερομηνιών.

Όταν ο χρήστης πατήσει το κουμπί BOOK NOW όπως φαίνεται στην κάρτα στην Εικόνα 14 θα οδηγηθεί στην σελίδα της θέσης στάθμευσης.



Test Spot 1

Vehicle Type: suv

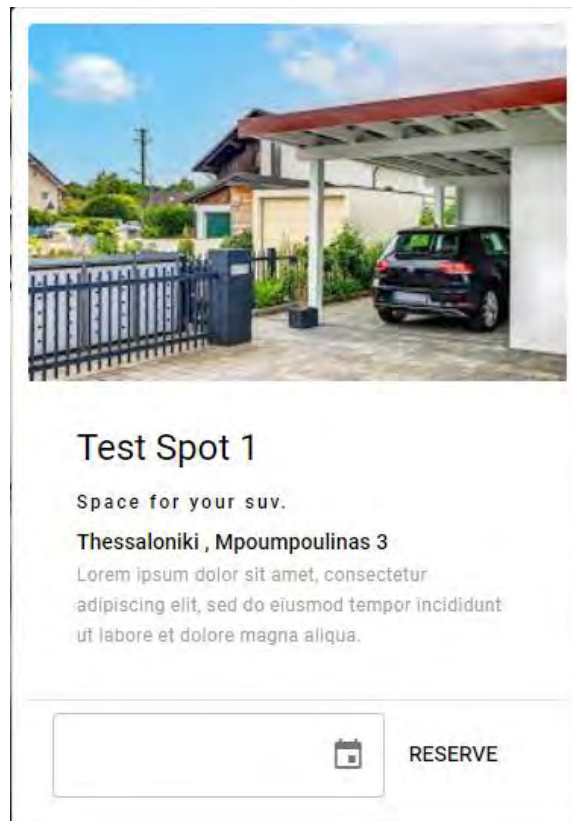
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

Thessaloniki, Μπουμπουλίνας 3

BOOK NOW

Εικόνα 14. Κάρτα θέσης στάθμευσης όπως εμφανίζεται στη λίστα

Παρατηρούμε στην Εικόνα 15, ότι στην κάρτα θέσης στάθμευσης ο χρήστης έχει τη δυνατότητα να βρει κάποιες πληροφορίες για τη θέση. Στο κάτω κομμάτι υπάρχει ένα κενό κουτί στο οποίο μπορεί να πατήσει το εικονίδιο με το ημερολόγιο και να ορίσει ημερομηνίες για να εξασφαλίσει τη θέση στάθμευσης. Αυτό το επιτυγχάνει πατώντας το κουμπί RESERVE δεξιά από το κουμπί των ημερομηνιών.

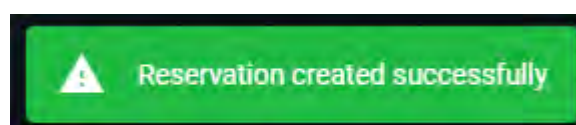


Εικόνα 15. Κάρτα θέσης στάθμευσης όπως εμφανίζεται στη σελίδα της

Αρχικά θα εμφανιστεί μία κάρτα επιβεβαίωσης (Εικόνα 16) και πατώντας ΟΚ όπως φαίνεται στην θα έχει καταφέρει να κάνει κράτηση για αυτή τη θέση. Με την ολοκλήρωση της κράτησης του εμφανίζεται ένα μήνυμα επιβεβαίωσης όπως φαίνεται στην Εικόνα 17.



Εικόνα 17. Κάρτα επιβεβαίωσης κράτησης.



Εικόνα 18. Μήνυμα επιτυχούς δημιουργίας θέσεως στάθμευσης

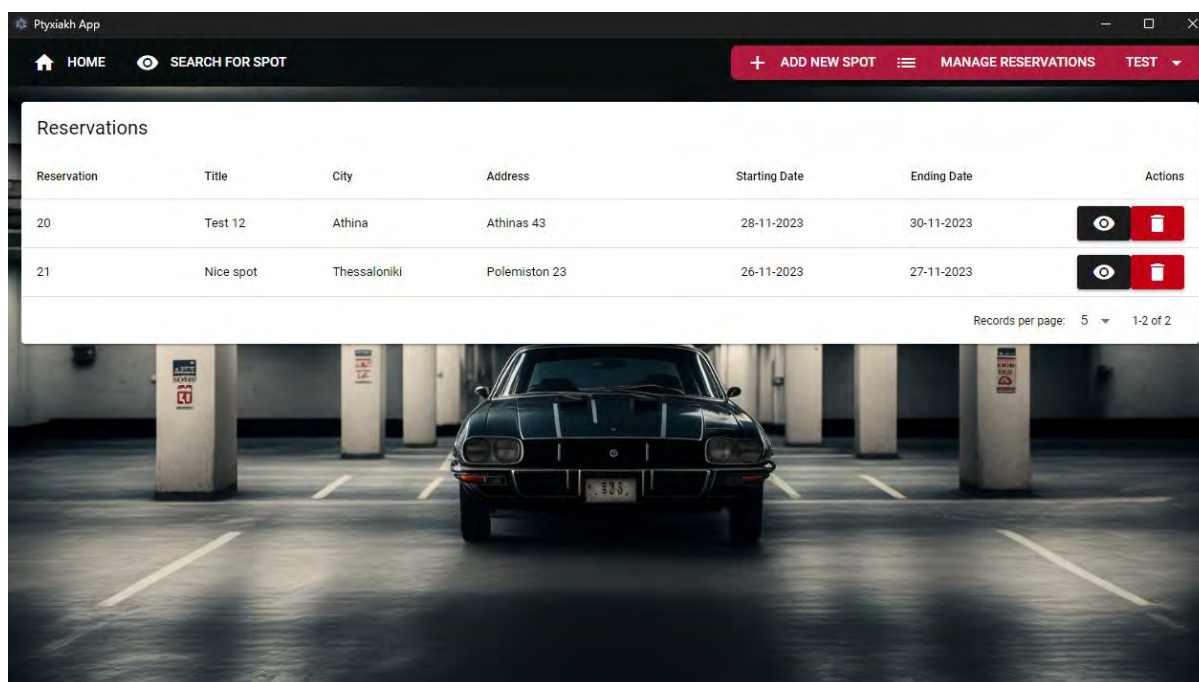
Στην παρακάτω λίστα ο χρήστης βλέπει πληροφορίες σε σχέση με τις κρατήσεις του.

Οι πληροφορίες είναι οι εξής:

- ο κωδικός της κράτησής
- το όνομα της θέσης κράτησης
- η πόλη
- η οδός
- οι ημερομηνίες που ξεκινάει και τελειώνει κράτηση.

Τέλος βρίσκει δύο κουμπιά:

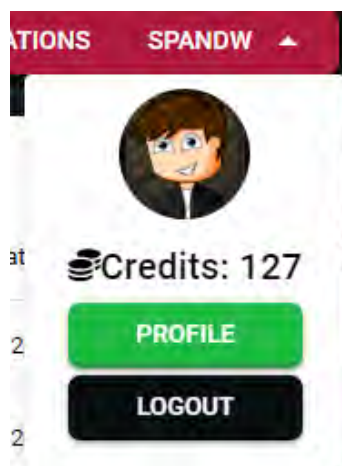
- ένα μαύρο κουμπί με εικονίδιο ματιού που μπορεί να επισκεφτεί τη σελίδα της θέσης
- ένα κόκκινο κουμπί με έναν κάδο απορριμμάτων που μπορεί να ακυρώσει την κράτηση



Εικόνα 19. Λίστα με ενεργές κρατήσεις του χρήστη

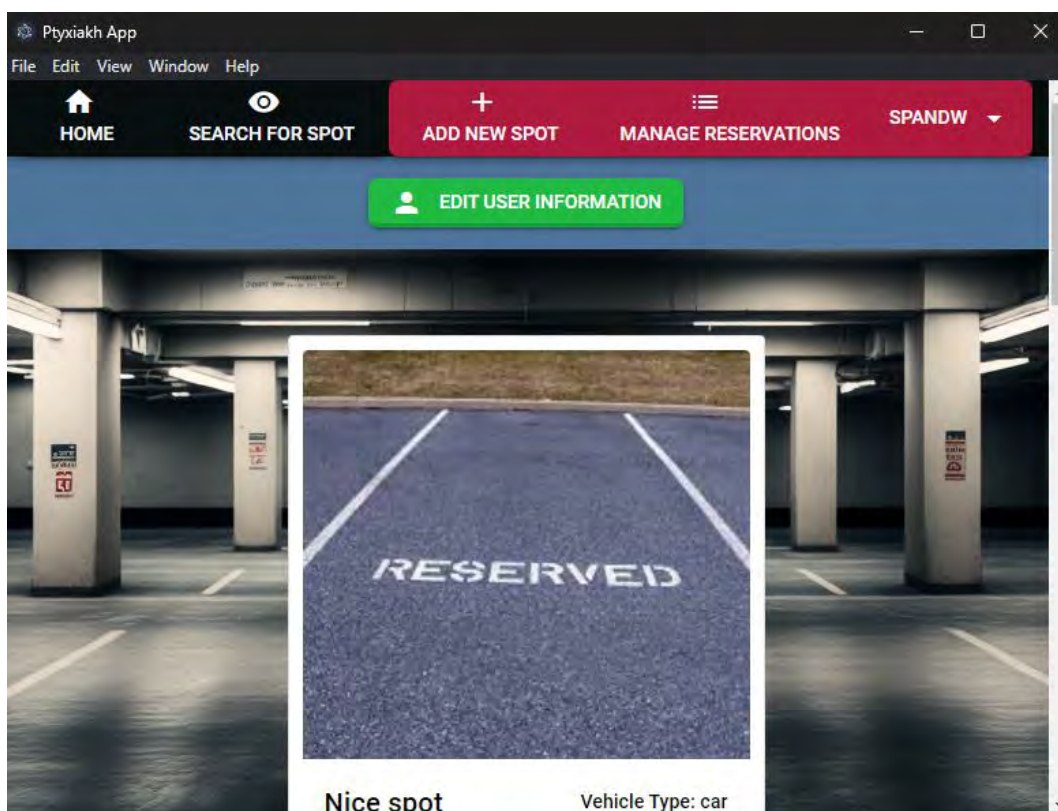
Σε περίπτωση που ο χρήστης πατήσει στο κουμπί με το username του εμφανίζεται η παρακάτω dropdown λίστα όπως φαίνεται στην Εικόνα 19:

- Ένα εικονίδιο νομισμάτων με τα διαθέσιμα credits του χρήστη
- Ένα πράσινο κουμπί profile, ο χρήστης μπορεί να επισκεφθεί το προφίλ του.
- Ένα μαύρο κουμπί logout, ο χρήστης μπορεί να αποσυνδεθεί από την εφαρμογή.



Εικόνα 20. Dropdown κάρτα με ενέργειες για τον χρήστη

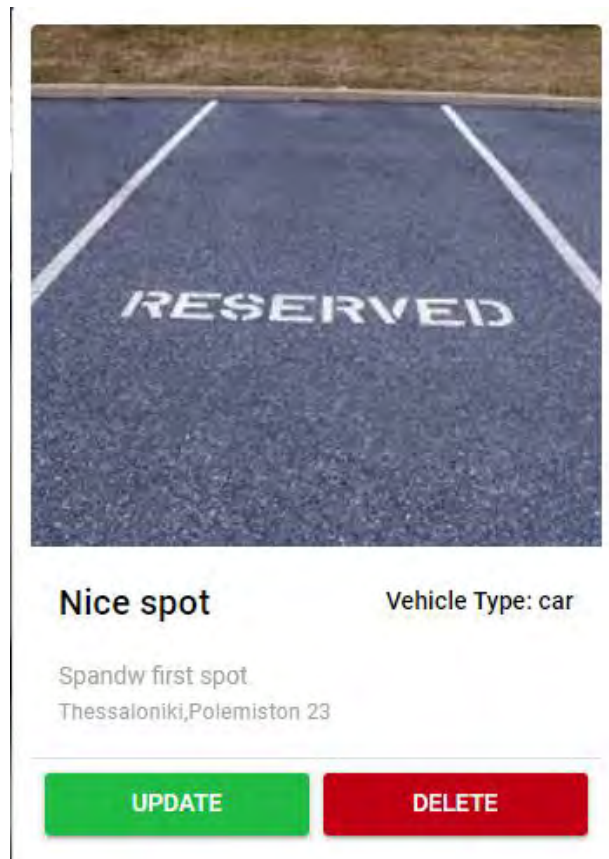
Αν ο χρήστης επισκεφθεί το προφίλ του θα αντικρίσει ένα πράσινο κουμπί που γράφει edit user information και εκεί μπορεί να αλλάξει τα στοιχεία του λογαριασμού του και από κάτω εμφανίζεται μία λίστα με τις θέσεις που είναι υπό την ιδιοκτησία του χρήστη όπως φαίνεται στην Εικόνα 20.



Εικόνα 21 Λίστα με τις θέσεις που ανήκουν στον χρήστη.

Στις κάρτες με τις θέσεις στάθμευσης που είναι υπό την ιδιοκτησία του χρήστη, όπως παρατηρούμε στην Εικόνα 21, υπάρχουν δύο κουμπιά:

- Ένα πράσινο κουμπί update και είναι για την ενημέρωση των πληροφοριών της συγκεκριμένης θέσης στάθμευσης.
- Ένα κόκκινο κουμπί που γράφει delete και είναι για τη διαγραφή της συγκεκριμένης θέσης στάθμευσης από την εφαρμογή.

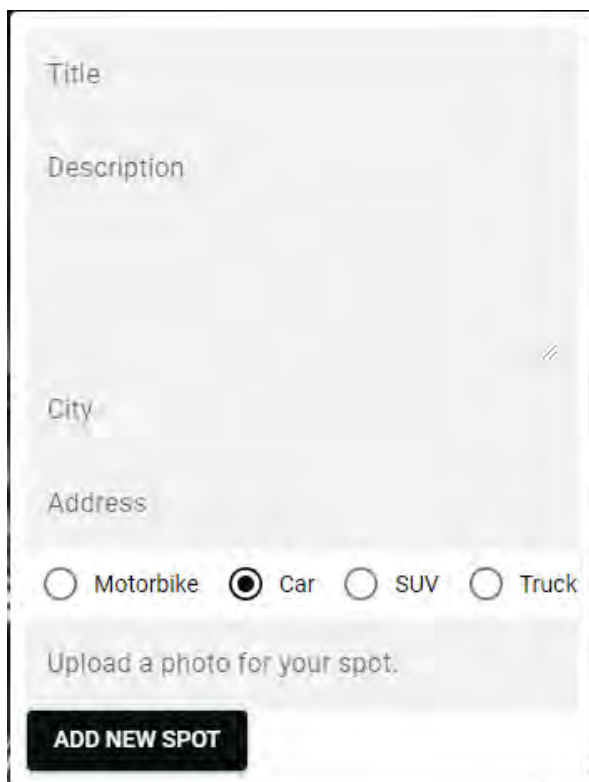


Εικόνα 22. Κάρτα με θέση που ανήκει στο χρήστη.

Σε περίπτωση που χρήστης πατήσει το κουμπί add new spot τότε θα εμφανιστεί η κάρτα που βλέπουμε στην Εικόνα 23 και του ζητείται να συμπληρώσει:

- ένα τίτλο που να περιγράφει την θέση στάθμευσης
- μία αναλυτικότερη περιγραφή της θέσης
- την πόλη
- τη διεύθυνση
- να επιλέξει τον τύπο του οχήματος
- μία φωτογραφία από τη θέση

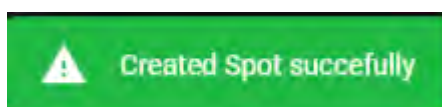
Τέλος ο χρήστης μπορεί πατήσει το κουμπί add new spot και θα του εμφανιστεί ένα παράθυρο επιβεβαίωσης όπως φαίνεται στην Εικόνα 24 και μπορεί να πατήσει OK έτσι ώστε να ολοκληρώσει την διαδικασία. Αν όλες οι πληροφορίες είναι σωστές η θέση στάθμευσης θα έχει ανέβει στην εφαρμογή και θα εμφανιστεί το αντίστοιχο μήνυμα επιτυχούς καταχώρησης όπως φαίνεται στην Εικόνα 25.



Εικόνα 23. Κάρτα προθήκης νέας θέσης στάθμευσης.

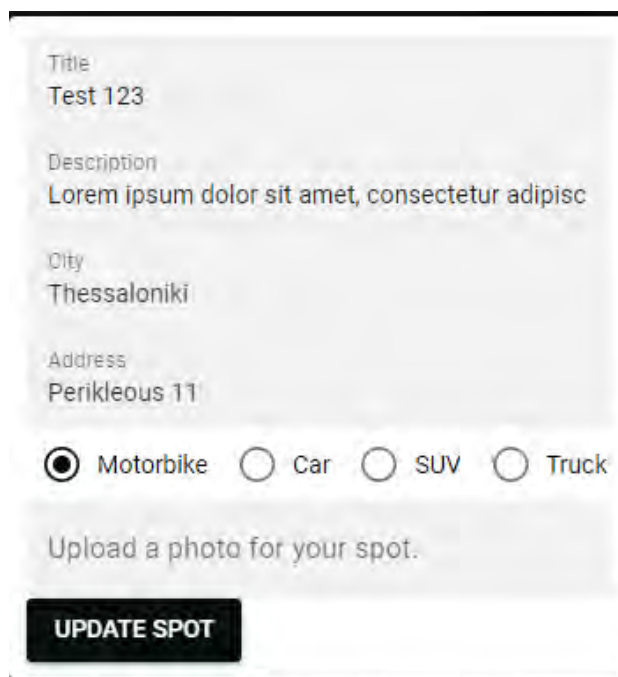


Εικόνα 24. Κάρτα επιβεβαίωσης δημιουργίας νέας θέσης στάθμευσης



Εικόνα 25. Μήνυμα επιτυχούς δημιουργίας νέας θέσης στάθμευσης

Στην περίπτωση που ο χρήστης πατήσει το πράσινο κουμπί που γράφει update θα του εμφανιστεί μία κάρτα με τις υπάρχουσες πληροφορίες για τη θέση και θα μπορεί να τροποποιήσει όποια πληροφορία επιθυμεί. Όταν πατήσει το κουμπί update spot οι πληροφορίες θα ενημερωθούν και θα αλλάξουν στη βάση δεδομένων.



Εικόνα 26. Κάρτα για την ενημέρωση πληροφοριών υπάρχουσας θέσης στάθμευσης.

ΚΕΦΑΛΑΙΟ 5 Συμπεράσματα

Το πρόβλημα της αναζήτησης και εύρεσης θέσης στάθμευσης είναι ένα πρόβλημα που επηρεάζεται από πολλούς συντελεστές. Με την χρήση των framework και την δημιουργία μια υβριδικής εφαρμογής καταφέραμε να δώσουμε μια νέα πρόταση για την βοήθεια της επίλυσης του προβλήματος. Πιο αναλυτικά αυτή η εφαρμογή καταφέρνει:

- Να βελτιστοποιήσει τους χώρους στάθμευσης: Με αυτή την εφαρμογή προσθέσαμε στις διαθέσιμες θέσεις στάθμευσης, όλες τις ιδιωτικές θέσεις που μέχρι πρότινος εξυπηρετούσαν μόνο τα οχήματα του ιδιοκτήτη της.
- Να προσφέρει ευκολία στους χρήστες: Οι χρήστες μέσα σε λίγα λεπτά έχουν την δυνατότητα να αναζητήσουν και να κλείσουν μια θέση στάθμευσης κοντά στον προορισμό τους, χωρίς το άγχος για εύρεση θέσης εκείνη την ώρα. Συμπεραίνουμε, ότι η εφαρμογή έχει βελτιώσει πραγματικά τη συνολική εμπειρία στάθμευσης για τους χρήστες της.

- Οικονομικό αντίκτυπο για τους χρήστες: Από τη στιγμή που εφαρμογή είναι δωρεάν και η ανταλλαγή των θέσεων δεν πραγματοποιείται με χρηματικό αντίτιμο, η εφαρμογή συνέβαλε σε μια πιο οικονομική προσέγγιση για τη στάθμευση σε αστικά περιβάλλοντα.

Προτάσεις για μελλοντική ανάπτυξη

- Χάρτες: Η προσθήκη χαρτών στην εφαρμογή θα μπορούσε να διευκολύνει τους χρήστες να βρουν διαθέσιμες θέσεις στάθμευσης κοντά τους. Οι χάρτες θα μπορούσαν να απεικονίζουν τις θέσεις στάθμευσης που είναι διαθέσιμες.
- Οικονομικό μοντέλο: Η ανάπτυξη ενός οικονομικού μοντέλου για την εφαρμογή θα μπορούσε να βοηθήσει να προσδιοριστεί και να προβλεφθεί ο τρόπος με τον οποίο θα μπορούσε να είναι βιώσιμη. Το μοντέλο θα μπορούσε να εξετάσει παράγοντες όπως το κόστος ανάπτυξης και συντήρησης της εφαρμογής, καθώς και τα έσοδα που θα μπορούσαν να δημιουργηθούν από χρεώσεις για χρήση ή από διαφημίσεις.
- Ανάπτυξη αντίστοιχης υβριδικής εφαρμογής για κινητό τηλέφωνο: Ο τρόπος που υλοποιήθηκε η εφαρμογή επιτρέπει σε ανάπτυξη εφαρμογής για κινητό τηλέφωνο με τον ήδη υπάρχοντα κώδικα και με τον περαιτέρω εμπλουτισμό του.

Βιβλιογραφία

Βιβλία

- [1] Φραντζεσκάκης I., M.X Πιτσιάβα-Λατινοπούλου, Δ.Α Τσαμπούλας, 2002. *Στάθμευση*. Αθήνα: Εκδόσεις Παπασωτηρίου
- [2] Rahul Raj, C. ., & Seshu Babu Tolety. (2012). *A study on approaches to build cross-platform mobile applications and criteria to select appropriate approach*.
- [3] Prokofyeva N and Boltunova V 2017 *Analysis and Practical Application of PHP Frameworks in Development of Web Information Systems Procedia Computer Science*.
- [4] Laaziri M, Benmoussa K, Khouilji S and Kerkeb M L 2019 *A Comparative study of PHP frameworks performance Procedia Manufacturing*
- [5] P. R. M. de Andrade, A. B. Albuquerque, O. F. Frota, R. V Silveira, and F. A. da Silva. *Cross platform app: a comparative study*. ArXiv e-prints, March 2015.

[6] Chen, X., Ji, Z., Fan, Y., & Zhan, Y. (2017). *Restful API Architecture Based on Laravel Framework. Journal of Physics: Conference Series*

Ηλεκτρονικές Πηγές

- [7] <https://www.php.net/docs.php>
- [8] <https://laravel.com/docs/8.x>
- [9] <https://vuejs.org/>
- [10] <https://www.electronjs.org/docs/latest/>
- [11] <https://quasar.dev/>
- [12] <https://datatracker.ietf.org/doc/html/rfc7159>
- [13] https://developer.mozilla.org/en-US/docs/Learn/JavaScript/First_steps/What_is_JavaScript
- [14] <https://dev.mysql.com/doc/>
- [15] <https://axios-http.com/docs/intro>
- [16] <https://developer.mozilla.org/en-US/docs/Web/HTML>
- [17] <https://developer.mozilla.org/en-US/docs/Web/CSS>
- [18] <https://pinia.vuejs.org/introduction.html>
- [19] <https://www.electronjs.org/docs/latest/>

Παράρτημα-Πηγαίος Κώδικας

Παράρτημα Α

Για την εγκατάσταση και εκτέλεση της Laravel:

- `composer global require laravel/installer`
- `laravel new example-app`
- `cd example-app`
- `php artisan serve`

Για την εγκατάσταση και εκτέλεση Quasar:

- `npm i -g @quasar/cli`
- `npm init quasar`
- `quasar dev`

Για την εγκατάσταση και εκτέλεση `electron.js`:

- `mkdir my-electron-app && cd my-electron-app`
- `npm init`
- `npm install --save-dev electron`
- `npm start`

Παράρτημα Β Πηγαίος Κώδικας

Main.js electron

```
const { app, BrowserWindow } = require('electron')
const createWindow = () => {
  const win = new BrowserWindow({
    width: 1280,
    height: 720
  })
  win.setMenuBarVisibility(false)
  win.loadURL('http://localhost:8080/')
}
app.whenReady().then(() => {
  createWindow()
})
```

MainLayout.vue

```
<template>
  <q-layout view="lHh Lpr lFf">
    <q-header elevated v-if="userStore.loggedIn">
      <q-toolbar>
        <q-btn-group push>
          <q-btn push label="Home" icon="home" to="/" />
          <q-btn
            push
            label="Search For Spot"
            icon="visibility"
            to="/parking-spots"
          />
        </q-btn-group>
        <q-space />
        <q-btn-group push>
          <q-btn
            color="accent"
            icon="add"
            label="Add New Spot"
          />
        </q-btn-group>
      </q-toolbar>
    </q-header>
  </q-layout>
```

```

    to="/create-spot"
  />
  <q-btn
    color="accent"
    icon="lists"
    label="Manage Reservations"
    to="/user-reservations"
  />
  <q-btn-dropdown color="accent" :label="userStore.user.username">
    <div class="row no-wrap q-pa-sm">
      <div class="column items-center">
        <q-avatar size="72px">
          
        </q-avatar>
        <q-list dense padding>
          <q-item>
            <div class="row items-center justify-between">
              <i class="fa-solid fa-coins fa-lg"></i>
              <div class="text-h6">Credits: {{ user.credits }}</div>
            </div>
          </q-item>
          <q-item>
            <q-item-section>
              <q-btn
                color="positive"
                label="Profile"
                push
                size="m"
                v-close-popup
                to="/profile"
              />
            </q-item-section>
          </q-item>
          <q-item>
            <q-item-section>
              <q-btn
                color="primary"
                label="Logout"
                push
                size="m"
                v-close-popup
                @click="userStore.logout()"
              />
            </q-item-section>
          </q-item>
        </q-list>
      </div>
    </div>
  </q-btn-dropdown>

```

```

        </div>
      </q-btn-dropdown>
    </q-btn-group>
  </q-toolbar>
</q-header>
<q-page-container>
  <Suspense>
    <!-- component with nested async dependencies -->
    <router-view />
    <!-- loading state via #fallback slot -->
    <template #fallback>
      <div>Loading...</div>
    </template>
  </Suspense>
</q-page-container>
</q-layout>
</template>

<script setup>
import { storeToRefs } from "pinia";
import { useParkingSpotStore } from "src/stores/ParkingSpotStore";
import { useUserStore } from "src/stores/UserStore";

const userStore = useUserStore();
const parkingSpotStore = useParkingSpotStore();
const { user } = storeToRefs(userStore);

userStore.loadUser();
</script>

```

ParkingSpotCard.vue

```

<template padding>
  <div class="container">
    <q-card class="my-card" flat bordered>
      <q-img :src="imageLink" height="300px" />
      <q-card-section>
        <div class="row no-wrap items-center">
          <div class="col text-h6 ellipsis">{{ spot.title }}</div>
          <div class="text-subtitle2">
            Vehicle Type: {{ spot.vehicle_type }}
          </div>
        </div>
      </q-card-section>

      <q-card-section class="q-pt-none">
        <div class="text-paragraph text-grey">
          {{ spot.description }}
        </div>
      </q-card-section>
    </q-card>
  </div>
</template>

```

```

    <q-space />
    <div class="text-caption text-grey">
      {{ spot.city }},{{ spot.address }}
    </div>
  </q-card-section>
  <q-separator />
  <q-card-actions class="row">
    <q-btn
      filler
      v-if="!checkOwnership"
      class="col"
      color="primary"
      :to="`/parking-spot/${spot.id}`"
    >
      Book now
    </q-btn>
    <q-btn
      filler
      v-if="checkOwnership"
      class="col"
      color="positive"
      :to="`/update-spot/${spot.id}`"
    >
      Update
    </q-btn>
    <q-btn
      filler
      v-if="checkOwnership"
      class="col"
      color="negative"
      @click="deleteSpot(spot.id)"
    >
      Delete
    </q-btn>
  </q-card-actions>
</q-card>
</div>
</template>

<script setup>
import { useQuasar } from "quasar";
import { useParkingSpotStore } from "src/stores/ParkingSpotStore";
import { useUserStore } from "src/stores/UserStore";
import { computed } from "vue";

const userStore = useUserStore();
const parkingSpotStore = useParkingSpotStore();
const $q = useQuasar();

```

```

const checkOwnership = computed(() => userStore.user.id ===
props.spot.user_id);

const imageLink = computed(
  () => `http://127.0.0.1:8000/storage/parking-spots/${props.spot.id}.png`
);

const props = defineProps({
  spot: {
    type: Object,
    required: true,
  },
});
const deleteSpot = (id) => {
  $q.dialog({
    title: "Delete Parking Spot",
    message: "Are you sure you want to delete this parking spot?",
    cancel: true,
    persistent: true,
  })
  .onOk(() => {
    console.log(id);

    parkingSpotStore.deleteSpot(id);
    parkingSpotStore.getUserParkingSpots(userStore.user.id);
  })
  .onCancel(() => {
    // console.log('>>>> Cancel')
  })
  .onDismiss(() => {
    // console.log('I am triggered on both OK and Cancel')
  });
};
</script>

<style lang="sass" scoped>
.container
  max-width: 400px
  margin: 15px
</style>

```

SingleSpotPage.vue

```

<template>
  <q-page padding class="flex flex-center items-center">
    <q-card class="my-card" v-if="parkingSpot" flat bordered>
      <q-img :src="imageLink" />
    <q-card-section>

```

```

<q-card-section class="q-pt-xs">
  <div class="text-h5 q-mt-sm q-mb-xs">{{ parkingSpot.title }}</div>
  <div class="text-overline">
    Space for your {{ parkingSpot.vehicle_type }}.
  </div>

  <div class="text-subtitle2">
    {{ parkingSpot.city }} , {{ parkingSpot.address }}
  </div>
  <div class="text-caption text-grey">
    {{ parkingSpot.description }}
  </div>
</q-card-section>
</q-card-section>
<q-separator />
<q-card-actions>
  <q-input
    bg-color="white"
    outlined
    :model-value="stringDate"
    class="q-mx-sm"
  >
    <template v-slot:append>
      <q-icon name="event" class="cursor-pointer">
        <q-popup-proxy
          cover
          transition-show="scale"
          transition-hide="scale"
        >
          <q-date
            v-model="date"
            minimal
            range
            :options="filteredDateOptions"
            @update:model-value="validateDate"
            mask="YYYY-MM-DD"
          >
            <div class="row items-center justify-end">
              <q-btn v-close-popup label="Close" color="primary" flat />
            </div>
          </q-date>
        </q-popup-proxy>
      </q-icon>
    </template>
  </q-input>
  <q-btn flat color="primary" @click="reserveSpot"> Reserve </q-btn>
</q-card-actions>
</q-card>

```

```

    </q-page>
</template>

<script setup>
import { storeToRefs } from "pinia";
import { useQuasar } from "quasar";
import ParkingSpotCard from "src/components/ParkingSpotCard.vue";
import { useParkingSpotStore } from "src/stores/ParkingSpotStore";
import { useUserStore } from "src/stores/UserStore";
import { computed, ref } from "vue";
import { useRoute, useRouter } from "vue-router";

const route = useRoute();
const parkingSpotStore = useParkingSpotStore();
const userStore = useUserStore();
const { parkingSpot, spotReservations } = storeToRefs(parkingSpotStore);
const stringDate = ref("");
const date = ref({
  from: "",
  to: "",
});
const $q = useQuasar();

const imageLink = computed(
  () =>
    `http://127.0.0.1:8000/storage/parking-spots/${parkingSpot.value.id}.png`
);

const id = ref(route.params.id);
parkingSpotStore.fetchParkingSpot(id.value);
await parkingSpotStore.getParkingSpotReservations(id.value);

console.log("apo to component: ", spotReservations.value);

const validateDate = () => {
  console.log(
    "date.value.from",
    date.value.from,
    "-",
    "date.value.to",
    date.value.to,
    "date.value",
    date.value
  );
  if (date.value.from) {
    stringDate.value = `${date.value.from} - ${date.value.to}`;
  } else {
    stringDate.value = date.value;
  }
}

```

```

    }
  };

  const filteredDateOptions = (date) => {
    return !spotReservations.value?.includes(date);
  };

  const reserveSpot = () => {
    $q.dialog({
      title: "Reserve Parking Spot",
      message: "Are you sure you want to rereserve this parking spot?",
      cancel: true,
      persistent: true,
    }).onOk(() => {
      parkingSpotStore.reserveParkingSpot({
        parking_spot_id: id.value,
        start_date: date.value.from,
        end_date: date.value.to,
      });
      parkingSpotStore.getParkingSpotReservations(id.value);
      userStore.loadUser();
    });
  };
</script>

<style lang="sass" scoped>
.my-card
  width: 100%
  max-width: 60vh
</style>

```

ParkingSpotList.vue

```

<template>
  <q-page>
    <div
      class="fit row wrap justify-evenly content-start q-pa-md"
      style="margin-top: 100px"
    >
      <ParkingSpotCard
        v-for="(parking, index) in filteredSpots"
        :key="index"
        :spot="parking"
      />
    </div>
    <q-page-sticky
      expand
      position="top"
      style="background-color: #4f759b; opacity: 100%"
    >

```

```

>
  <div
    class="flex fit row wrap justify-center items-start content-center q-
pa-md"
  >
    <q-select
      class="q-mx-sm"
      bg-color="white"
      outlined
      v-model="selectedCity"
      use-input
      hide-selected
      fill-input
      input-debounce="0"
      :options="filteredCities"
      @filter="filterFn"
      clearable
      placeholder="City"
    >
      <template v-slot:no-option>
        <q-item>
          <q-item-section class="text-grey"> No results </q-item-section>
        </q-item>
      </template>
    </q-select>
    <q-select
      class="q-mx-sm"
      bg-color="white"
      outlined
      v-model="selectedCarType"
      use-input
      hide-selected
      fill-input
      input-debounce="0"
      :options="carTypes"
      clearable
      placeholder="Vehicle Type"
    >
  </q-select>

  <q-input
    bg-color="white"
    outlined
    :model-value="stringDate"
    class="q-mx-sm"
    placeholder="Date"
  >
  <template v-slot:append>

```

```

        <q-icon name="event" class="cursor-pointer">
          <q-popup-proxy
            cover
            transition-show="scale"
            transition-hide="scale"
          >
            <q-date
              v-model="date"
              range
              minimal
              @update:model-value="validateDate"
              mask="YYYY-MM-DD"
            >
              <q-btn-group spread>
                <q-btn
                  v-close-popup
                  label="Search"
                  color="secondary"
                  @click="searchWithDates"
                />
                <q-btn label="Clear" color="negative" @click="clearDate"
              />
            </q-date>
          </q-popup-proxy>
        </q-icon>
      </template>
    </q-input>
  </div>
</q-page-sticky>
</q-page>
</template>

<script setup>
import { storeToRefs } from "pinia";
import { useQuasar } from "quasar";
import ParkingSpotCard from "src/components/ParkingSpotCard.vue";
import { useParkingSpotStore } from "src/stores/ParkingSpotStore";
import { useUserStore } from "src/stores/UserStore";
import { computed, ref } from "vue";

const parkingSpotStore = useParkingSpotStore();
const userStore = useUserStore();

const { cities, parkingSpotList } = storeToRefs(parkingSpotStore);

await parkingSpotStore.getAllParkingSpots();

```

```

parkingSpotStore.getUniqueCities();
//const today = new Date();
const selectedCity = ref(null);
const selectedCarType = ref(null);
const filteredCities = ref(null);

const stringDate = ref("");
const date = ref({
  from: "",
  to: "",
});

const carTypes = ["motorbike", "car", "suv", "truck"];

const validateDate = () => {
  console.log(
    "date.value.from",
    date.value.from,
    "-",
    "date.value.to",
    date.value.to,
    "date.value",
    date.value
  );
  if (date.value.from) {
    stringDate.value = `${date.value.from} - ${date.value.to}`;
  } else {
    stringDate.value = date.value;
  }
};

const searchWithDates = () => {
  parkingSpotStore.filterParkingSpotByDate(date);
};

const clearDate = () => {
  stringDate.value = "";
  date.value = { from: "", to: "" };
};

const filterFn = (val, update) => {
  update(() => {
    const needle = val.toLowerCase();
    filteredCities.value = cities.value.filter((v) => {
      return v.toLowerCase().indexOf(needle) > -1;
    });
  });
};

```

```

const filteredSpots = computed(() => {
  return parkingSpotList.value.filter((parkingSpot) => {
    return (
      (parkingSpot.city === selectedCity.value ||
        selectedCity.value === null) &&
      (selectedCarType.value === parkingSpot.vehicle_type ||
        selectedCarType.value === null) &&
      parkingSpot.user_id !== userStore.user.id
    );
  });
});
</script>

```

ParkingSpotStore.js

```

import { defineStore } from "pinia";
import { ref } from "vue";
import { useRouter } from "vue-router";
import axios from "axios";
import { useUserStore } from "../UserStore";
import { route } from "quasar/wrappers";
import { useQuasar } from "quasar";

export const useParkingSpotStore = defineStore("parkingSpotStore", () => {
  const router = useRouter();
  const userStore = useUserStore();
  const $q = useQuasar();

  const parkingSpot = ref(null);
  const parkingSpotList = ref(null);
  const cities = ref([]);
  const spotReservations = ref([]);
  const userReservations = ref([]);

  const getAllParkingSpots = async () => {
    await axios
      .get("http://127.0.0.1:8000/api/parking-spots")
      .then((response) => {
        console.log("Get all parking spots(store): ", response.data);
        parkingSpotList.value = response.data;
        //console.log(response);
      })
      .catch(function (error) {
        // handle error
        console.log(error);
      });
  };

  const getUserParkingSpots = async (userId) => {
    await axios

```

```

    .get(`http://127.0.0.1:8000/api/parking-spots/${userId}`)
    .then((response) => {
      console.log(userStore.user.id);
      console.log(response.data);
      parkingSpotList.value = response.data;
      //console.log(response);
    })
    .catch(function (error) {
      // handle error
      console.log(error);
    });
  });

const getUserReservations = async () => {
  await axios
    .get("http://127.0.0.1:8000/api/user-reservations")
    .then((response) => {
      userReservations.value = response.data;
      console.log("sto store: ", userReservations.value);
    })
    .catch(function (error) {
      // handle error
      console.log(error);
    });
};

const fetchParkingSpot = async (id) => {
  console.log("psaxnw gia spot");
  await axios
    .get(`http://127.0.0.1:8000/api/parking-spot/${id}`)
    .then((response) => {
      console.log(response.data);
      parkingSpot.value = response.data;
      //console.log(response);
    })
    .catch(function (error) {
      // handle error
      console.log(error);
    });
};

const getUniqueCities = () => {
  axios
    .get("http://127.0.0.1:8000/api/cities")
    .then((response) => {
      console.log(response.data);
      cities.value = response.data;
      //console.log(response);
    })

```

```

        .catch(function (error) {
            // handle error
            console.log(error);
        });
    });

const createNewSpot = async (data) => {
    return axios
        .post("http://127.0.0.1:8000/api/create-spot", data)
        .then((response) => {
            $q.notify({
                color: "positive",
                message: response.data.message,
                icon: "report_problem",
            });
            router.push({ path: "/" });
        })
        .catch((error) => {
            $q.notify({
                color: "negative",
                message: error.response.data.message,
                icon: "report_problem",
            });
        });
    });

const updateParkingSpot = async (data) => {
    console.log("data: ", data);
    return axios
        .post(`http://127.0.0.1:8000/api/update-spot/${data.id}`, data)
        .then(
            $q.notify({
                color: "positive",
                message: response.data.message,
                icon: "report_problem",
            })
        );
    });

const deleteSpot = async (spotId) => {
    try {
        const response = await axios.delete(
            `http://127.0.0.1:8000/api/delete-spot/${spotId}`
        );
        $q.notify({
            color: "positive",
            message: response.data.message,
            icon: "report_problem",
        });
    }

```

```

    });
    router.push({ path: "/profile" });
  } catch (error) {
    $q.notify({
      color: "negative",
      message: error.response.data.message,
      icon: "report_problem",
    });
  }
};

const getParkingSpotReservations = async (spotId) => {
  await axios
    .get(`http://127.0.0.1:8000/api/check/${spotId}`)
    .then((response) => {
      console.log("sto store: ", response.data);
      spotReservations.value = response.data;
      //console.log(response);
    })
    .catch(function (error) {
      // handle error
      console.log(error);
    });
};

const filterParkingSpotByDate = async (date) => {
  await axios
    .get(
      `http://127.0.0.1:8000/api/check?start_date=${
        date.value.from ?? date.value
      }&end_date=${date.value.to ?? ""}`
    )
    .then((response) => {
      console.log("Reservations Object:", response.data);
      parkingSpotList.value = response.data.available_spots;
      console.log(
        "New Spot List(after date filter): ",
        parkingSpotList.value
      );
    })
    .catch(function (error) {
      // handle error
      console.log(error);
    });
};

const reserveParkingSpot = async (data) => {
  try {

```

```

const response = await axios.post(
  "http://127.0.0.1:8000/api/create-reservation",
  data
);
$q.notify({
  color: "positive",
  message: response.data.message,
  icon: "report_problem",
});
router.push({ path: "/" });
} catch (error) {
  $q.notify({
    color: "negative",
    message: error.response.data.message,
    icon: "report_problem",
  });
}
};

const cancelReservation = async (id) => {
  try {
    const response = await axios.delete(
      `http://127.0.0.1:8000/api/cancel-reservation/${id}`
    );
    $q.notify({
      color: "warning",
      message: response.data.message,
      icon: "report_problem",
    });
  } catch (error) {
    $q.notify({
      color: "negative",
      message: error.response.data.message,
      icon: "report_problem",
    });
  }
};

return {
  cities,
  parkingSpotList,
  parkingSpot,
  spotReservations,
  userReservations,
  getUniqueCities,
  getAllParkingSpots,
  getUserParkingSpots,
  createNewSpot,

```

```

    updateParkingSpot,
    fetchParkingSpot,
    deleteSpot,
    getParkingSpotReservations,
    filterParkingSpotByDate,
    reserveParkingSpot,
    getUserReservations,
    cancelReservation,
  };
});

```

routes.js

```

import LoginPage from "pages/LoginPage";
import RegisterPage from "pages/RegisterPage";
import LandingPage from "pages/LandingPage";
import ParkingSpotList from "pages/ParkingSpotList";
import CreateNewSpot from "pages/CreateNewSpot";
import UpdateExistingSpot from "pages/UpdateExistingSpot";
import UserProfile from "pages/UserProfile";
import EditUser from "pages/EditUser";
import SingleSpotPage from "pages/SingleSpotPage";
import UserReservationList from "pages/UserReservationList";

const routes = [
  {
    path: "/",
    component: () => import("layouts/MainLayout.vue"),
    children: [
      {
        path: "",
        component: () => import("pages/IndexPage.vue"),
        meta: { requiresAuth: true },
      },
      {
        path: "login",
        component: LoginPage,
        meta: { requiresAuth: false },
      },
      {
        path: "register",
        component: RegisterPage,
        meta: { requiresAuth: false },
      },
      {
        path: "landing",
        component: LandingPage,
        meta: { requiresAuth: false },
      },
    ],
  },

```

```

    {
      path: "parking-spots",
      component: ParkingSpotList,
      meta: { requiresAuth: true },
      children: [],
    },
    {
      path: "user-reservations",
      component: UserReservationList,
      meta: { requiresAuth: true },
      children: [],
    },
    {
      path: "parking-spot/:id",
      component: SingleSpotPage,
      meta: { requiresAuth: true },
    },
    {
      path: "create-spot",
      component: CreateNewSpot,
      meta: { requiresAuth: true },
    },
    {
      path: "update-spot/:id",
      component: UpdateExistingSpot,
      meta: { requiresAuth: true },
    },
    {
      path: "profile",
      component: UserProfile,
      meta: { requiresAuth: true },
    },
    {
      path: "edit-user",
      component: EditUser,
      meta: { requiresAuth: true },
    },
  ],
},
// Always leave this as last one,
// but you can also remove it
{
  path: "/*:catchAll(.*)*",
  component: () => import("pages/ErrorNotFound.vue"),
},
];

```

```
export default routes;
```

api.php

```
<?php

use Illuminate\Http\Request;
use Illuminate\Support\Facades\Route;
use App\Http\Controllers\ApiUserController;
use App\Http\Controllers\ApiLoginController;
use App\Http\Controllers\ApiParkingController;
use App\Http\Controllers\ApiReservationsController;

/*
|-----
| API Routes
|-----
|
| Here is where you can register API routes for your application. These
| routes are loaded by the RouteServiceProvider within a group which
| is assigned the "api" middleware group. Enjoy building your API!
|
*/

Route::post('/register', [ApiLoginController::class, 'register']);
Route::post('/login', [ApiLoginController::class, 'authenticate']);

Route::group(['middleware' => ['auth:sanctum']], function () {

    Route::get('/check', [ApiReservationsController::class,
'checkAvailability']);
    Route::get('/check/{id}', [ApiReservationsController::class,
'getSpotReservationsDates']);
    Route::post('/create-reservation', [ApiReservationsController::class,
'createReservation']);
    Route::delete('/cancel-reservation/{id}',
[ApiReservationsController::class, 'destroy']);

    Route::post('/create-spot', [ApiParkingController::class, 'store']);
    Route::post('/update-spot/{id}', [ApiParkingController::class, 'update']);
    Route::delete('/delete-spot/{id}', [ApiParkingController::class,
'destroy']);
    Route::get('/cities', [ApiParkingController::class, 'getDistinctCities']);
    Route::get('/parking-spots', [ApiParkingController::class,
'getFilteredParkingSpots']);
    Route::get('/parking-spots/{id}', [ApiUserController::class,
'getUserParkingSpots']);
    Route::get('/parking-spot/{id}', [ApiParkingController::class,
'getParkingSpotById']);
});
```

```

Route::get('/user/{id}', [ApiUserController::class, 'getUserById']);
Route::get('/users', [ApiUserController::class, 'getAllUsers']);
Route::put('/edit-user', [ApiUserController::class, 'update']);
Route::get('/user-reservations', [ApiUserController::class,
'getUserReservations']);

Route::get('/user', [ApiLoginController::class, 'getCurrentUser']);
Route::post('/logout', [ApiLoginController::class, 'logout']);
});

```

ApiParkingController.php

```

<?php

namespace App\Http\Controllers;

use App\Models\ParkingSpot;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\App;
use Illuminate\Support\Facades\Storage;

class ApiParkingController extends Controller
{
    public function store(Request $request)
    {
        $data = $request->validate([
            'city' => ['required', 'string'],
            'address' => ['required', 'string'],
            'title' => ['required', 'string'],
            'description' => ['required', 'string'],
            'type' => 'in:motorbike,car,suv,truck',
            'photo' => ['string'],
        ]);

        $parkingSpot = ParkingSpot::create([

            'city' => $data['city'],
            'address' => $data['address'],
            'title' => $data['title'],
            'description' => $data['description'],
            'vehicle_type' => $data['type'],
            'user_id' => $request->user()->id
        ]);

        $base64_image = $data['photo'];

        if (preg_match('/^data:image\/(\w+);base64,/', $base64_image)) {

```

```

        $data = substr($base64_image, strpos($base64_image, ',') + 1);

        $data = base64_decode($data);
        Storage::disk('public')->put("/parking-spots/$parkingSpot-
>id.png", $data);
    }

    return response()->json([
        'message' => 'Created Spot succefully',
        'parking_spot' => $parkingSpot

    ], 200);
}

public function update(Request $request, $id)
{
    $data = $request->validate([
        'city' => ['required', 'string'],
        'address' => ['required', 'string'],
        'title' => ['required', 'string'],
        'description' => ['required', 'string'],
        'vehicle_type' => 'in:motorbike,car,suv,truck',
        'photo' => ['string'],

    ]);

    $parkingSpot = ParkingSpot::find($id);
    if (!$parkingSpot) {
        return response()->json([
            'message' => 'This Parking Spot Doesn\'t exist'
        ], 404);
    }

    $parkingSpot->city = $data['city'];
    $parkingSpot->address = $data['address'];
    $parkingSpot->title = $data['title'];
    $parkingSpot->description = $data['description'];
    $parkingSpot->vehicle_type = $data['vehicle_type'];

    $base64_image = $data['photo'];

    if (preg_match('/^data:image\/(\w+);base64/,', $base64_image)) {
        $data = substr($base64_image, strpos($base64_image, ',') + 1);

        $data = base64_decode($data);
        Storage::disk('public')->put("/parking-spots/$parkingSpot-
>id.png", $data);
    }
}

```

```

    }
    $parkingSpot->save();

    return response()->json([
        'parking_spot' => $parkingSpot
    ], 200);
}
public function destroy($id)
{
    $parkingSpot = ParkingSpot::find($id);
    $parkingSpot->delete();

    return response()->json([
        'message' => "Parking spot deleted successfully!"
    ], 200);
}

public function getFilteredParkingSpots(Request $request)
{
    $query = ParkingSpot::query();
    // Apply filters based on request parameters
    if ($request->has('vehicle_type')) {
        $query->where('vehicle_type', $request->input('vehicle_type'));
    }
    if ($request->has('city')) {
        $query->where('city', $request->input('city'));
    }

    $parkingSpots = $query->get();
    return response()->json($parkingSpots);
    if (!$parkingSpots) {
        return response()->json([
            'message' => 'There are no parking spots yet'
        ], 404);
    }
    return response()->json($parkingSpots, 200);
}

public function getParkingSpotById($id)
{
    $parkingSpot = ParkingSpot::find($id);
    if (!$parkingSpot) {
        return response()->json([
            'message' => 'There are no parking spots yet'
        ], 404);
    }
    return response()->json($parkingSpot, 200);
}

```

```

public function getDistinctCities()
{
    $cities = ParkingSpot::distinct()->pluck('city');
    if (!$cities) {
        return response()->json([
            'message' => 'There are no cities'
        ], 404);
    }
    return response()->json($cities, 200);
}
}

```

ApiReservationsController.php

```

<?php

namespace App\Http\Controllers;

use Carbon\Carbon;
use App\Models\ParkingSpot;
use App\Models\Reservation;
use App\Models\User;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\DB;

class ApiReservationsController extends Controller
{
    public function getSpotReservationsDates($id)
    {
        $reservations = Reservation::query()
            ->where('parking_spot_id', $id)
            ->select('start_date', 'end_date')
            ->get();

        $formattedDates = $this->formatDateRanges($reservations);

        return response()->json($formattedDates);
    }

    private function formatDateRanges($dateRanges)
    {
        $formattedDates = [];

        foreach ($dateRanges as $range) {
            $startDate = Carbon::parse($range['start_date']);
            $endDate = Carbon::parse($range['end_date']);

```

```

        while ($startDate <= $endDate) {
            // Format the date as "YYYY/MM/DD" (date only)
            $formattedDates[] = $startDate->format('Y/m/d');
            $startDate->addDay();
        }
    }

    return $formattedDates;
}

public function checkAvailability(Request $request)
{
    $check_start_date = $request->start_date;
    $check_end_date = $request->end_date;

    $reservations = Reservation::checkDates($check_start_date,
    $check_end_date)->pluck('parking_spot_id');

    $available_spots = ParkingSpot::whereNotIn('id', $reservations)
    ->get();

    return response()->json([
        'message' => "Reservations on the dates given.",
        'reservations' => $reservations,
        'available_spots' => $available_spots
    ], 200);
}

public function createReservation(Request $request)
{
    $check_start_date = $request->start_date;
    $check_end_date = $request->end_date;

    if (empty($request->end_date)) {
        $check_start_date = $request->date;
        $check_end_date = $request->date;
    }
    //dates se int
    $int_start_date = Carbon::parse($check_start_date);
    $int_end_date = Carbon::parse($check_end_date);

    $daysDifference = $int_end_date->diffInDays($int_start_date);

    $user_id = auth('sanctum')->user()->id;
    $reservee = User::findOrFail($user_id);

```

```

        $isNotAvailable = Reservation::checkDates($check_start_date,
$check_end_date)
        ->where('parking_spot_id', $request->parking_spot_id)->exists());
    if ($isNotAvailable) {
        return response()->json([
            'message' => "There is already a reservation on those dates."
        ], 422);
    }

    $reserveeCredits = $reservee->credits;
    if ($reserveeCredits < $daysDifference) {
        return response()->json([
            'message' => "You have insufficient amount of credits",
            'Your Credits' => $reserveeCredits,
            'days Difference' => $daysDifference
        ], 422);
    }
    $parkingSpot = ParkingSpot::find($request->parking_spot_id);
    $owner_id = $parkingSpot->user_id;

    $owner = User::find($owner_id);
    $ownerCredits = $owner->credits + $daysDifference;
    $owner->update(['credits' => $ownerCredits]);

    $reserveeCredits = $reserveeCredits - $daysDifference;
    User::where('id', $user_id)->update(['credits' => $reserveeCredits]);

    $reservation = Reservation::create([
        'user_id' => $user_id,
        'parking_spot_id' => $request->parking_spot_id,
        'start_date' => $check_start_date,
        'end_date' => $check_end_date,
    ]);

    return response()->json([
        'message' => "Reservation created successfully",
        'reservation' => $reservation,
    ], 200);
}

public function destroy($id)
{
    $reservation = Reservation::find($id);
    $reservation->delete();

    return response()->json([
        'message' => "Reservation was cancelled!"
    ], 200);
}

```

}

}