ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ – ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Πρόγραμμα Μεταπτυχιακών Σπουδών «Προηγμένα Συστήματα Πληροφορικής – Ανάπτυξη Λογισμικού και Τεχνητής Νοημοσύνης»

Μεταπτυχιακή Διατριβή

Τίτλος Διατριβής	**ΑΝΑΠΤΥΞΗ ΕΦΑΡΜΟΓΗΣ ANDROID ΜΕ ΑΝΤΙΚΕΙΜΕΝΟ ΤΗΝ ΑΝΑΖΗΤΗΣΗ ΚΑΙ ΕΝΟΙΚΙΑΣΗ ΙΔΙΩΤΙΚΩΝ ΧΩΡΩΝ ΣΤΑΘΜΕΥΣΗΣ** **ANDROID APPLICATION DEVELOPMENT FOR SEARCHING AND RENTING PERSONAL PARKING SPACES**
Ονοματεπώνυμο Φοιτητή	**ΘΕΟΔΩΡΟΥ ΑΓΓΕΛΙΚΗ**
Πατρώνυμο	**ΚΩΝΣΤΑΝΤΙΝΟΣ**
Αριθμός Μητρώου	**ΜΠΣΠ 21017**
Επιβλέπων	**ΑΛΕΠΗΣ ΕΥΘΥΜΙΟΣ, ΑΝΑΠΛΗΡΩΤΗΣ ΚΑΘΗΓΗΤΗΣ**

Ημερομηνία παράδοσης **Οκτώβριος 2023**

Τριμελής Εξεταστική Επιτροπή

Ευθύμιος Αλέπης	Μαρία Βίρβου	Κωνσταντίνος Πατσάκης
Αναπληρωτής Καθηγητής	Καθηγήτρια	Αναπληρωτής Καθηγητής

Πρόλογος

Διανύοντας μια εποχή κατά την οποία η τεχνολογία, η οικονομία και ο εκσυγχρονισμός των πόλεων εξελίσσονται συνεχώς και με ραγδαίους ρυθμούς, η κυκλοφοριακή συμφόρηση και η αναζήτηση θέσεων στάθμευσης αποτελούν σημαντικά κοινωνικά προβλήματα. Η έλλειψη ισορροπίας μεταξύ της προσφοράς και της ζήτησης των θέσεων παρκινγκ είναι η κυριότερη πηγή των παραπάνω προβλημάτων ενώ παράλληλα αποτελεί σημαντικό παράγοντα πρόκλησης κυκλοφοριακών ατυχημάτων, περιβαλλοντικής ρύπανσης και επηρεάζει την καθημερινότητα των οδηγών.

Η εργασία αυτή πραγματεύεται την ανάπτυξη μιας εφαρμογής με την οποία ο χρήστης μπορεί να ενοικιάσει το δικό του προσωπικό χώρο πάρκινγκ - όποτε αυτός είναι διαθέσιμος - ως λύση στο πρόβλημα ανεύρεσης χώρων στάθμευσης, αυξάνοντας έτσι τις πιθανές θέσεις πάρκινγκ που υπάρχουν σε μια περιοχή για κάθε οδηγό. Σκοπός του λογισμικού αυτού είναι η εφαρμογή των δυνατοτήτων της πληροφορικής και των τηλεπικοινωνιών για την αύξηση του παράγοντα της προσφοράς στο φαινομενικά άλυτο πρόβλημα της προσφοράς – ζήτησης των θέσεων στάθμευσης στις αστικές περιοχές. Αυτό επιτυγχάνεται με τη δημιουργία ενός περιβάλλοντος εύκολου στη χρήση, το οποίο προσφέρει τη δυνατότητα στο χρήστη να επιβλέπει σε πραγματικό χρόνο τις διαθέσιμες θέσεις που μπορεί να σταθμεύσει, αλλά και να ενοικιάζει τον προσωπικό του χώρο στάθμευσης.

Abstract

Navigating an era where technology, economy, and urban modernization are continually advancing at rapid rates, traffic congestion and the availability of parking spaces constitute significant societal issues. The imbalance between parking supply and demand is the primary source of the aforementioned problems. Simultaneously, it stands as a crucial factor in traffic accidents and environmental pollution, while also impacting drivers' daily routines.

This assignment addresses the development of an application through which users can rent their own personal parking space whenever it's available, presenting a solution to the problem of parking space scarcity. This approach increases the potential parking spots available in an area for every driver. The software's purpose lies in applying the capabilities of information technology and telecommunications to augment the supply factor in the seemingly unsolvable issue of parking supply and demand in urban areas. This achievement is accomplished by creating a user-friendly environment that allows users to monitor real-time available parking spaces and rent their own parking space when needed.

Πίνακας περιεχομένων

Κατάλογος Εικόνων.....5
ΚΕΦΑΛΑΙΟ 1ο: ΕΙΣΑΓΩΓΗ.....7
ΚΕΦΑΛΑΙΟ 2ο: ΑΝΑΣΚΟΠΗΣΗ ΠΕΔΙΟΥ.....8
ΚΕΦΑΛΑΙΟ 3ο: ΑΝΑΛΥΣΗ ΑΠΑΙΤΗΣΕΩΝ.....13
4.1 Σκοπός και Χρησιμότητα.....13
4.2 Απαιτήσεις Λογισμικού.....15
ΚΕΦΑΛΑΙΟ 4ο: ΑΡΧΙΤΕΚΤΟΝΙΚΗ ΚΑΙ ΤΕΧΝΟΛΟΓΙΕΣ.....17
5.1 Αρχιτεκτονική.....17
5.2 Τεχνολογίες.....20
5.2.1 Backend component.....20
5.2.2 Mobile component.....26
ΚΕΦΑΛΑΙΟ 5ο: ΥΛΟΠΟΙΗΣΗ ΕΦΑΡΜΟΓΗ.....31
6.1 Δομή Εφαρμογής.....31
6.1.1 Back-End.....31
6.1.2 Mobile.....36
ΚΕΦΑΛΑΙΟ 6ο: ΧΡΗΣΗ ΕΦΑΡΜΟΓΗΣ.....38
7.1 Εγκατάσταση.....38
7.2 Εγχειρίδιο Χρήστη.....40
7.2.1 Σύνδεση/Εγγραφή.....40
7.2.2 Χάρτης/Αναζήτηση θέσης στάθμευσης.....43
7.2.3 Προσθήκη νέας θέσης στάθμευσης.....44
7.2.4 Προβολή πληροφοριών θέσης στάθμευσης.....45
7.2.5 Κράτηση θέσης στάθμευσης.....46
7.2.6 Εισερχόμενα.....47
7.2.7 Ενοικιάσεις.....48
7.2.8 Ρυθμίσεις.....49
ΚΕΦΑΛΑΙΟ 7ο: ΣΥΜΠΕΡΑΣΜΑΤΑ ΚΑΙ ΜΕΛΛΟΝΤΙΚΕΣ ΕΠΕΚΤΑΣΕΙΣ.....54
8.1 Μελλοντικές Επεκτάσεις.....54
8.2 Συμπεράσματα.....55
ΒΙΒΛΙΟΓΡΑΦΙΑ.....56

Κατάλογος Εικόνων

Εικόνα 1 JustPark παράδειγμα 1 9
Εικόνα 2 JustPark παράδειγμα 2 9
Εικόνα 3 YourParkingSpace παράδειγμα 1 10
Εικόνα 4 YourParkingSpace παράδειγμα 2 10
Εικόνα 5 Stashbee παράδειγμα 1 11
Εικόνα 6 Stashbee παράδειγμα 2 11
Εικόνα 7 Διάγραμμα Περίπτωσης 1: Ενοικιαστής 14
Εικόνα 8 Διάγραμμα Περίπτωσης 2: Οδηγός 14
Εικόνα 9 Ο κύκλος της ανάπτυξης λογισμικού 15
Εικόνα 10 Αρχιτεκτονική Συστήματος 17
Εικόνα 11 Λεπτομερής Αρχιτεκτονική λογισμικού 19
Εικόνα 12 Λογότυπο PostgreSQL 20
Εικόνα 13 Δομή κύριας βάσης δεδομένων 20
Εικόνα 14 Firebase Storage δομή αρχείων 21
Εικόνα 15 Λογότυπο Django Rest Framework 21
Εικόνα 16 MVT πρότυπο σχεδίασης 22
Εικόνα 17 Λογότυπο Uvicorn 23
Εικόνα 18 Παράδειγμα εκτέλεσης ASGI server και API 23
Εικόνα 19 Δομή σύνδεσης μέσω Google 24
Εικόνα 20 Λογότυπο Firebase Cloud Messaging 25
Εικόνα 21 Δομή σύνδεσης εφαρμογής, server με το FCM 25
Εικόνα 22 Πρότυπο αρχιτεκτονικής MVVM 26
Εικόνα 23 Models 27
Εικόνα 24 User Model 27
Εικόνα 25 Views 28
Εικόνα 26 ViewModels 28
Εικόνα 27 Τμήμα κώδικα API requests 29
Εικόνα 28 Τμήμα κώδικα Firebase 29
Εικόνα 29 Λειτουργία Web Socket 30
Εικόνα 30 Διάγραμμα σύνδεσης Django Rest Framework 31
Εικόνα 31 Δομή εφαρμογής στο Visual Code 32
Εικόνα 32 Διάγραμμα δικτύου API και βάσης δεδομένων 33
Εικόνα 33 Διάγραμμα κλάσεων API 35
Εικόνα 34 Δομή mobile στο Android Studio 36
Εικόνα 35 Εκτέλεση βάσης, API και ασύγχρονου server 38

Εικόνα 36 Εφαρμογή ParkShare σε Android Emulator παράδειγμα 1 39
Εικόνα 37 Εφαρμογή ParkShare σε Android Emulator παράδειγμα 2 39
Εικόνα 38 ParkShare: Εισαγωγική οθόνη 40
Εικόνα 39 ParkShare: Οθόνη σύνδεσης 40
Εικόνα 40 ParkShare: Οθόνη Εγγραφής 41
Εικόνα 41 ParkShare: Συμπλήρωση στοιχείων προφίλ 41
Εικόνα 42 ParkShare: Οθόνη σύνδεσης, σφάλμα 42
Εικόνα 43 ParkShare: Οθόνη συμπλήρωσης προφίλ, σφάλμα 1 42
Εικόνα 44 ParkShare: Οθόνη συμπλήρωσης προφίλ, σφάλμα 2 42
Εικόνα 45 ParkShare: Βασική οθόνη, Χάρτης 43
Εικόνα 46 ParkShare: Βασική οθόνη, Φίλτρα 43
Εικόνα 47 ParkShare: Βασική οθόνη, Αναζήτηση Παρκινγκ 43
Εικόνα 48 ParkShare: Προσθήκη νέας θέσης πάρκινγκ, παράδειγμα 1 44
Εικόνα 49 ParkShare: Προσθήκη νέας θέσης πάρκινγκ, παράδειγμα 2 44
Εικόνα 50 ParkShare: Θέση πάρκινγκ στο χάρτη 45
Εικόνα 51 ParkShare: Πληροφορίες θέσης πάρκινγκ, Ενοικιαστής 45
Εικόνα 52 ParkShare: Πληροφορίες θέσης πάρκινγκ, Οδηγός 46
Εικόνα 53 ParkShare: Φόρμα κράτησης θέσης πάρκινγκ 46
Εικόνα 54 ParkShare: Πληροφορίες κράτησης 46
Εικόνα 55 ParkShare: Ειδοποίηση μηνύματος από την εφαρμογή 47
Εικόνα 56 ParkShare: Εισερχόμενα 47
Εικόνα 57 ParkShare: Συνομιλία 47
Εικόνα 58 ParkShare: Τρέχουσες Ενοικιάσεις, Οδηγοί 48
Εικόνα 59 ParkShare: Πληροφορίες κράτησης, Οδηγοί 48
Εικόνα 60 ParkShare: Ρυθμίσεις, Ενοικιαστές 49
Εικόνα 61 ParkShare: Ρυθμίσεις, Οδηγοί 49
Εικόνα 62 ParkShare: Οθόνη επεξεργασίας προφίλ 1 50
Εικόνα 63 ParkShare: Οθόνη επεξεργασίας προφίλ 2 50
Εικόνα 64 ParkShare: Οθόνη αλλαγής κωδικού πρόσβασης 50
Εικόνα 65 ParkShare: Αλλαγή γλώσσας 51
Εικόνα 66 ParkShare: Η εφαρμογή στα αγγλικά 51
Εικόνα 67 ParkShare: Τα παρκινγκ μου 52
Εικόνα 68 ParkShare: Διαγραφή παρκινγκ 52
Εικόνα 69 ParkShare: Τρέχουσες Ενοικιάσεις, Ενοικιαστές 53
Εικόνα 70 ParkShare: Προσεχείς Ενοικιάσεις, Ενοικιαστές 53
Εικόνα 71 ParkShare: Πληροφορίες κράτησης, Ενοικιαστής 53

ΚΕΦΑΛΑΙΟ 1°: ΕΙΣΑΓΩΓΗ

Η αναδόμηση της πολεοδομικής οργάνωσης των αστικών περιοχών θα μπορούσε ενδεχομένως να δώσει μια λύση στο πρόβλημα αναζήτησης χώρων στάθμευσης, ωστόσο για ευνόητους λόγους αποτελεί μια απίθανη επιλογή. Αντ' αυτού, η ανάπτυξη ηλεκτρονικών εφαρμογών στον αυτοκινητιστικό τομέα συμβάλλει στην επίλυση των προβλημάτων στάθμευσης και συστήνει σαν λύση τον όρο της *έξυπνης στάθμευσης* (smart parking). Χάρη στη ραγδαία εξέλιξη της πληροφορικής και των τηλεπικοινωνιών, οι οδηγοί έχουν τη δυνατότητα να εντοπίσουν με αποτελεσματικό τρόπο θέσεις πάρκινγκ με τη χρήση έξυπνων συσκευών. Αποκτώντας πρόσβαση στη διαθεσιμότητα πιθανών θέσεων στάθμευσης σε πραγματικό χρόνο, ένας οδηγός θα μπορούσε να προσαρμόσει πρόγραμμά του αντίστοιχα, προκειμένου να μη σπαταλήσει πολύτιμο χρόνο για την εύρεση χώρου για πάρκινγκ.

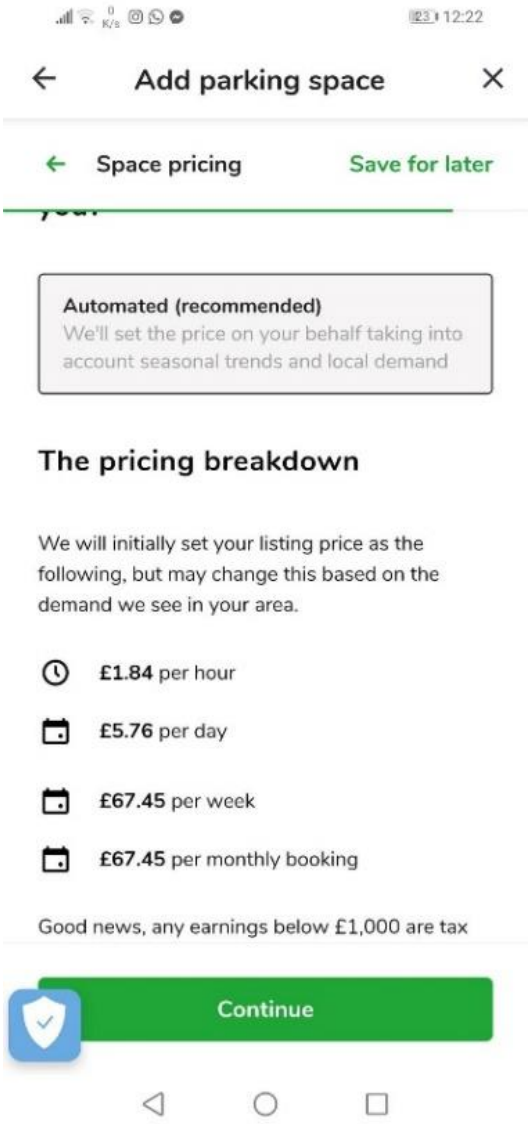
Καθώς η αύξηση των θέσεων στάθμευσης είναι αδύνατη, η μόνη λύση στο υπό μελέτη πρόβλημα είναι ο αποτελεσματικότερος εντοπισμός των ήδη υφιστάμενων διαθέσιμων θέσεων. Η ενοικίαση προσωπικών χώρων για πάρκινγκ ως εφαρμογή της «έξυπνης στάθμευσης» όχι μόνο προσφέρει θέσεις στάθμευσης στους οδηγούς αλλά και αυξάνει το οικονομικό κέρδος των ενοικιαστών. Μέσω της εφαρμογής αυτής οι οδηγοί μπορούν να ενημερώνονται εγκαίρως για κάθε διαθέσιμη θέση, να επικοινωνούν με τους ενοικιαστές και να κρατούν το χώρο που επιθυμούν άμεσα και εύκολα.

ΚΕΦΑΛΑΙΟ 2°: ΑΝΑΣΚΟΠΗΣΗ ΠΕΔΙΟΥ

Η χρήση εφαρμογών αναζήτησης και ενοικίασης προσωπικού χώρου στάθμευσης είναι μια γνωστή τακτική στο εξωτερικό. Συγκεκριμένα, στην Ευρώπη αλλά και στις Ηνωμένες Πολιτείες έχουν δημιουργηθεί και χρησιμοποιούνται παρόμοιες εφαρμογές, καθώς η ανάγκη ανεύρεσης θέσεων στάθμευσης και ειδικότερα σε οικονομικές τιμές είναι μονίμως αυξανόμενη. Στατιστικά, τα περισσότερα παρόμοια λογισμικά εφαρμόζονται στο Ηνωμένο Βασίλειο και ενδεικτικά κάποια παραδείγματα εξ αυτών είναι τα παρακάτω:

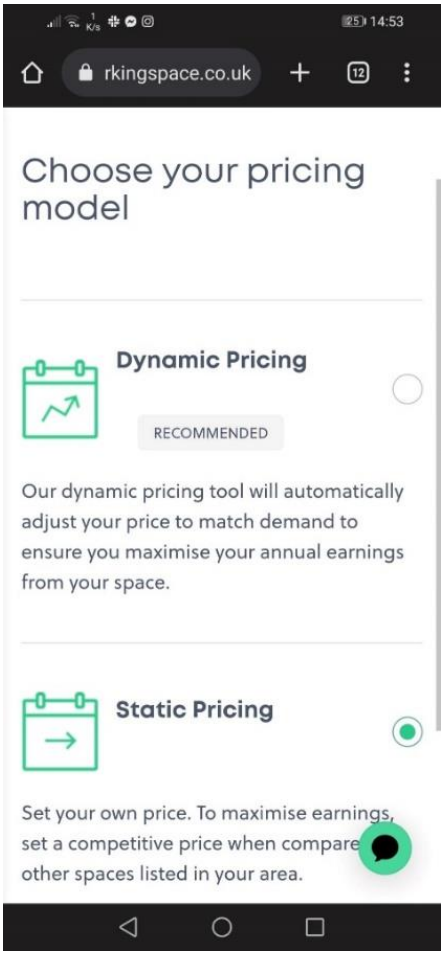
1. JustPark, μια εφαρμογή ευρέως χρησιμοποιούμενη κυρίως στο Ηνωμένο Βασίλειο (85.40%), στις Ηνωμένες Πολιτείες (3.57%), την Αυστραλία (1.08%), την Ινδία (0.88%), τη Γερμανία (0.84%) και άλλες χώρες (8.23%) για την υπηρεσία εντοπισμού χώρου στάθμευσης. Επιπρόσθετα, προσφέρει την υπηρεσία προσφοράς προς ενοικίαση προσωπικών χώρων στάθμευσης. Ο χρήστης μπορεί να δημοσιεύσει το προσωπικό του πάρκινγκ και το λογισμικό υπολογίζει αυτόματα την πλέον αποδοτική τιμή που μπορεί να κοστίζει η ενοικίαση, ανάλογα με την ανάλυση των τάσεων και της ζήτησης.

Εικόνα 1 JustPark παράδειγμα 1

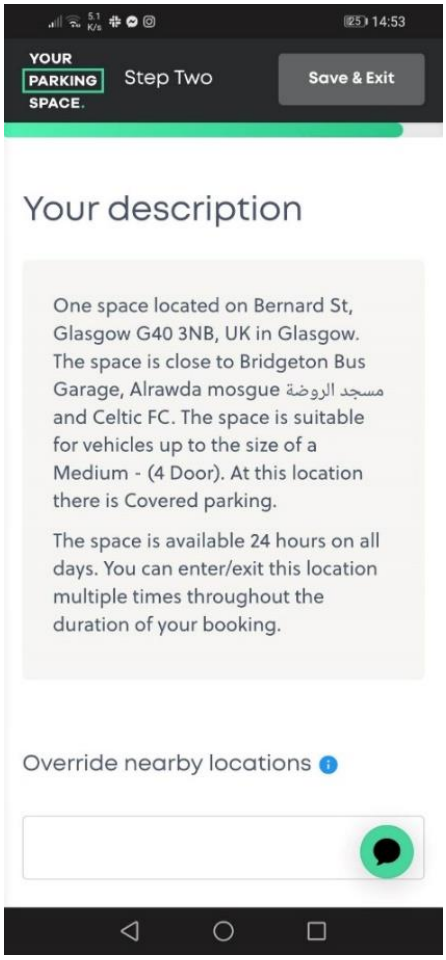


Εικόνα 2 JustPark παράδειγμα 2

2. YourParkingSpace, μια εφαρμογή που χρησιμοποιείται κυρίως στο Ηνωμένο Βασίλειο (97.05%), στις Ηνωμένες Πολιτείες (0.72%), την Ινδία (0.41%), το Βέλγιο (0.22%), την Γαλλία (0.18%) και άλλες χώρες (1.42%), για την υπηρεσία αναζήτησης πάρκινγκ αλλά και ενοικίασης προσωπικών χώρων στάθμευσης. Παρόμοια με την προαναφερθείσα εφαρμογή, ο χρήστης μπορεί να δημοσιεύσει το χώρο του αφού συμπληρώσει τις απαραίτητες πληροφορίες σχετικά με αυτόν και εφόσον εξεταστεί η εγκυρότητα των δεδομένων που υπέβαλλε. Το λογισμικό, αντίστοιχα, υπολογίζει με αυτόματο τρόπο την καταλληλότερη τιμή για τη θέση έπειτα από κάποια ανάλυση και παράγει ένα κείμενο που τη διαφημίζει για τους υπόλοιπους χρήστες.



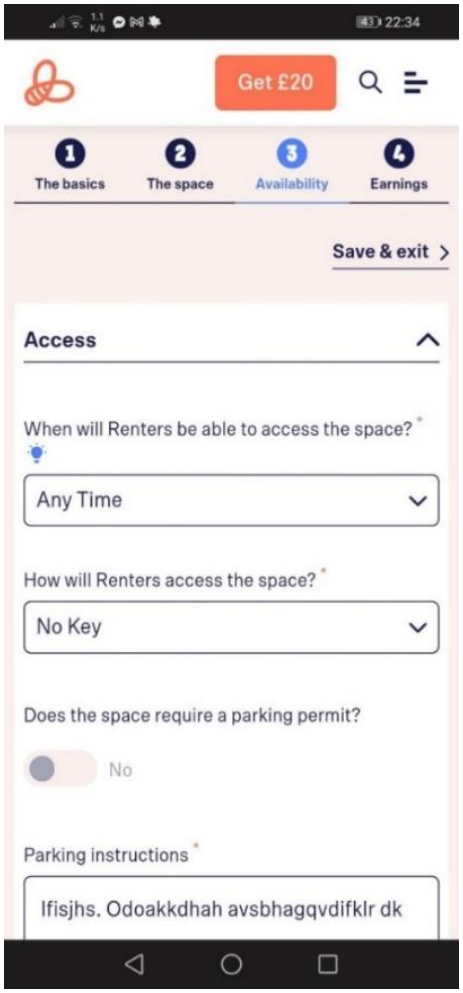
Εικόνα 3 YourParkingSpace παράδειγμα 1



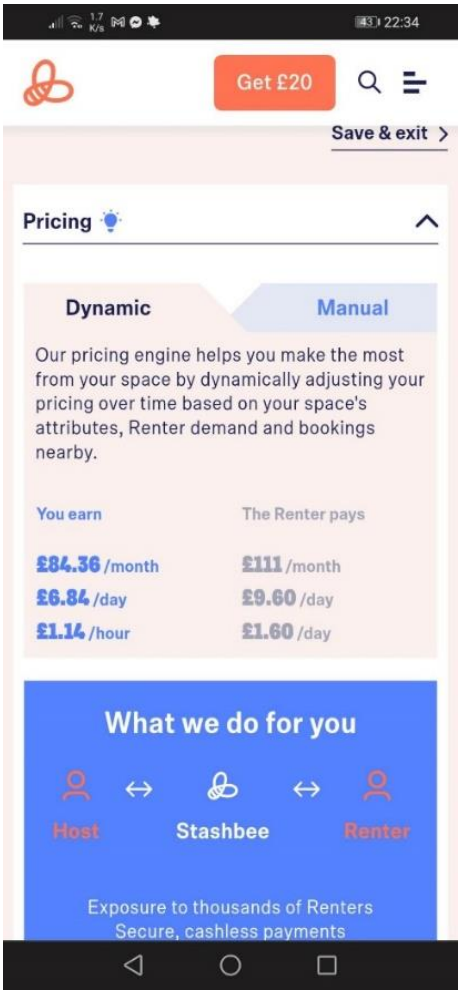
Εικόνα 4 YourParkingSpace παράδειγμα 2

3. Stashbee, μια εφαρμογή που χρησιμοποιείται κυρίως στο Ηνωμένο Βασίλειο (90.12%), στην Ελλάδα* (6.90%), τις Φιλιππίνες (1.09%), την Ταϊλάνδη (1.01%) και την Γαλλία (0.88%), για την υπηρεσία εντοπισμού χώρων για διαμονή και στάθμευση αλλά και για την υπηρεσία ενοικίασης προσωπικών χώρων διαμονής και στάθμευσης. Ο χρήστης μέσω της εφαρμογής αυτής μπορεί να δημοσιεύσει και να ενοικιάσει το χώρο του ενώ το λογισμικό του δίνει τη δυνατότητα υπολογισμού της κατάλληλης τιμής αυτόματα, όπως και στις προηγούμενες εφαρμογές.

* Είναι σημαντικό, να αναφερθεί πως η υπηρεσία ενοικίασης χώρου παρκινγκ δε λειτουργεί για την Ελλάδα.



Εικόνα 5 Stashbee παράδειγμα 1



Εικόνα 6 Stashbee παράδειγμα 2

Ενδεικτικά, αναφέρονται κάποια περαιτέρω παραδείγματα αντίστοιχων εφαρμογών, οι οποίες χρησιμοποιούνται κυρίως στο Ηνωμένο Βασίλειο και την Ευρώπη εκτός Ελλάδος: ParkOnMyDrive, ParkLet, Spacer, ParkPnp, ParkingForMe κ.α.

Είναι προφανές πως στην Ελλάδα πέραν των εφαρμογών που δίνουν τη δυνατότητα ενοικίασης προσωπικών χώρων για διαμονή, δε χρησιμοποιούνται εφαρμογές για την ενοικίαση χώρων για στάθμευση.

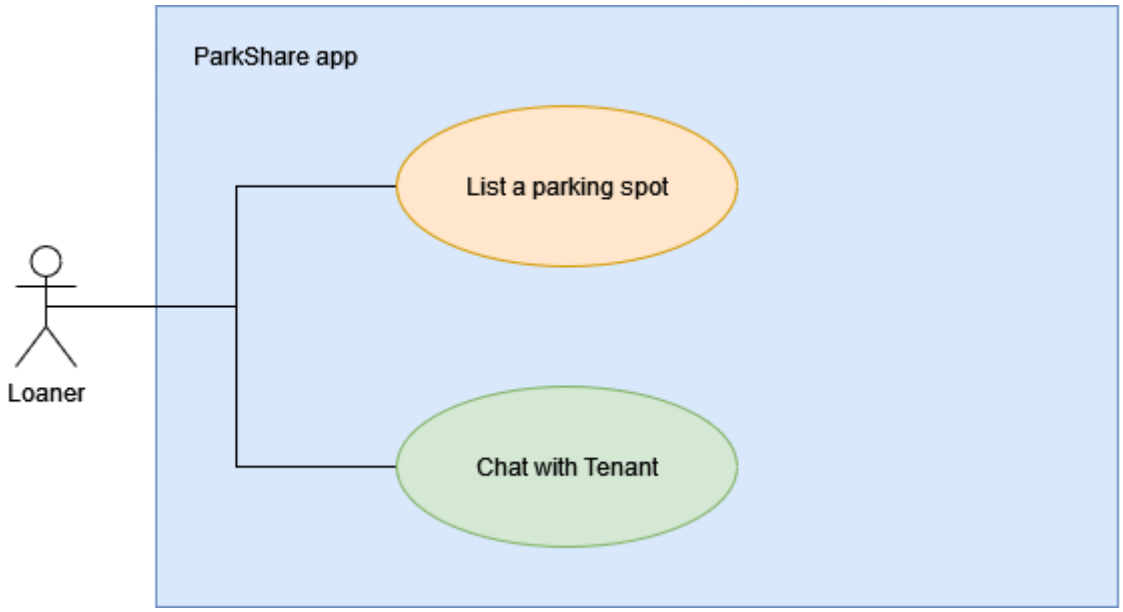
ΚΕΦΑΛΑΙΟ 3ο: ΑΝΑΛΥΣΗ ΑΠΑΙΤΗΣΕΩΝ

4.1 ΣΚΟΠΟΣ ΚΑΙ ΧΡΗΣΙΜΟΤΗΤΑ

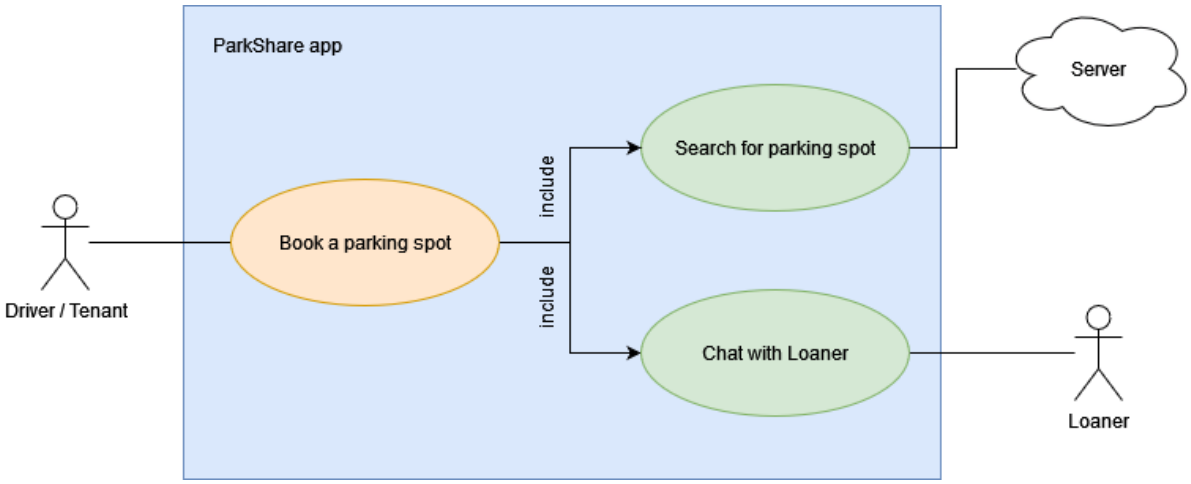
Η εργασία αυτή αποσκοπεί στη δημιουργία μιας προσωρινής λύσης στο πρόβλημα κορεσμού των θέσεων στάθμευσης, μέσω της ενοικίασης υφιστάμενων προσωπικών χώρων στάθμευσης. Συνεπώς, ως κύρια ομάδα ενδιαφερόμενων *(target group)* της εφαρμογής ορίζονται οι κοινοί οδηγοί που αναζητούν θέσεις παρκινγκ και οι πολίτες που επιθυμούν να ενοικιάσουν τους χώρους τους έτσι ώστε να εξασφαλίσουν κέρδος. Ωστόσο, προκειμένου να περιοριστεί η ομάδα των τελικών χρηστών και να διευκολυνθεί η διαδικασία της ανάλυσης απαιτήσεων, είναι σημαντικό να αναφερθεί πως χρήστες της παρούσας εφαρμογής μπορούν να είναι άτομα τα οποία είναι τεχνικά εξοικειωμένα με τη χρήση έξυπνων εφαρμογών σε κινητές συσκευές. Ακόμη, πέραν των οδηγών οι οποίοι αναζητούν θέσεις πάρκινγκ, η εφαρμογή αυτή μπορεί να φανεί χρήσιμη και σε άτομα τα οποία επιθυμούν να ενοικιάσουν το χώρο πάρκινγκ τους είτε είναι οι ίδιοι οδηγοί είτε όχι.

Όσον αφορά τις απαιτήσεις του λογισμικού είναι γνωστό πως στην ανάπτυξη κάθε λογισμικού η ανάλυση απαιτήσεων είναι από τα σημαντικότερα βήματα πριν την υλοποίηση, προκειμένου να σχεδιαστεί ένα ολοκληρωμένο εργαλείο *(Minimum Viable Product)*. Τις απαιτήσεις της συγκεκριμένης εργασίας ορίζουν οι παρακάτω κατηγορίες ως εξής:

1. Πρακτικές προϋποθέσεις
   Περιγράφουν τις λειτουργίες που χρειάζεται να υλοποιεί ένα εργαλείο. Αναφορικά με το παρόν λογισμικό, αυτές ορίζονται από τις βασικές υπηρεσίες που πρέπει να προσφέρει στους χρήστες, δηλαδή τη δυνατότητα αναζήτησης και ενοικίασης πάρκινγκ (εκ των ενοικιαζόμενων) και καταγραφή προσωπικού χώρου πάρκινγκ προς ενοικίαση. Για την ευκολότερη κατανόηση των πρακτικών απαιτήσεων σε συνδυασμό με τους ρόλους των χρηστών στο παρόν λογισμικό, παρατίθενται τα διαγράμματα περιπτώσεων χρήσης για τους Οδηγούς και τους Ενοικιαστές, αντίστοιχα.



Εικόνα 7 Διάγραμμα Περίπτωσης 1: Ενοικιαστής



Εικόνα 8 Διάγραμμα Περίπτωσης 2: Οδηγός

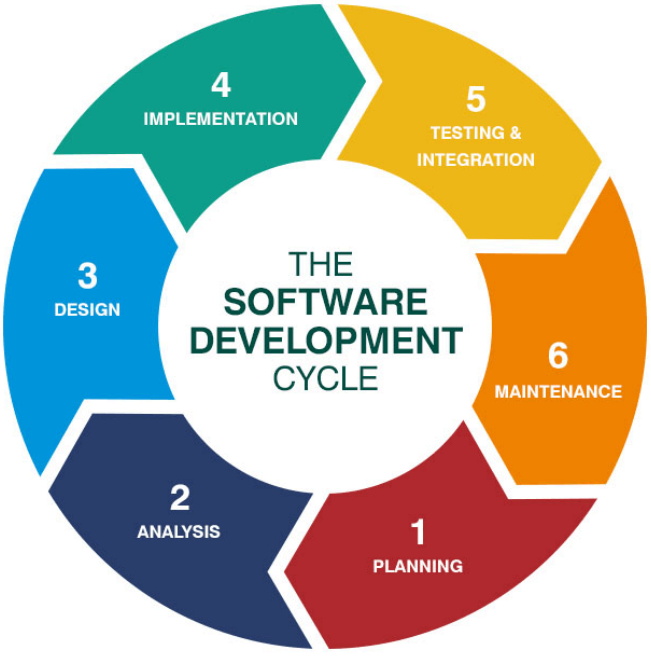
2. Τεχνικές προϋποθέσεις

Περιγράφουν τα βασικά τεχνικά θέματα τα οποία χρειάζεται να αναλυθούν προκειμένου να υλοποιηθεί το εργαλείο με επιτυχία. Αναφορικά με την παρούσα εφαρμογή, αυτά ορίζονται από το λειτουργικό σύστημα στο οποίο θα λειτουργεί το εργαλείο. Εφόσον αυτό θα χρησιμοποιείται από οδηγούς, είναι λογικό να είναι προσβάσιμο μέσω των πλέον διαδεδομένων κινητών συσκευών και έτσι μπορεί να υλοποιηθεί μέσω μιας Mobile (Android/iOS) εφαρμογής.

3. Λειτουργικές προϋποθέσεις
   Περιγράφουν τις διαδικασίες που θα πρέπει να γίνουν με στόχο τη σωστή λειτουργία του εργαλείου.

4.2 ΑΠΑΙΤΗΣΕΙΣ ΛΟΓΙΣΜΙΚΟΥ

Η διαδικασία ανάπτυξης λογισμικών αποτελείται από τέσσερα βασικά βήματα. Το πρώτο αφορά στην προετοιμασία, η οποία ανάγεται στην ανάλυση απαιτήσεων, προκειμένου να αποσαφηνιστεί η χρησιμότητα και οι στόχοι του εργαλείου. Το δεύτερο αφορά στην υλοποίηση και ανάγεται σε όλη την τεχνική εργασία που γίνεται για την ανάπτυξη του εργαλείου. Το τρίτο αφορά στον έλεγχο και την εγκυροποίηση του υπό ανάπτυξη λογισμικού. Αποτελεί μια πολύ σημαντική φάση στην ανάπτυξη λογισμικού καθώς μέσω αυτής αποφεύγονται διάφορα λάθη που μπορεί να προκύψουν στη συνέχεια, ενώ παράλληλα εδραιώνεται η εγκυρότητα της λειτουργίας της εφαρμογής. Τέλος, το τέταρτο βήμα αφορά στη δημοσίευση και στην διατήρηση του λογισμικού μακροπρόθεσμα, όπου γίνεται η εγκατάσταση του εργαλείου και η χρήση του από τους χρήστες ως αυτόνομη εφαρμογή.



Εικόνα 9 Ο κύκλος της ανάπτυξης λογισμικού

Για τις ανάγκες της παρούσας εργασίας ως προς τις τεχνικές απαιτήσεις, αρχικά έγινε μια κατηγοριοποίηση και έπειτα μια ενδεικτική ιεράρχηση των τεχνικών τμημάτων *(components)* που θα υλοποιούνταν στη συνέχεια. Τα βασικά τεχνικά μέρη της εφαρμογής χωρίζονται ως εξής:

Back-End components

Στον προκείμενο τεχνικό τομέα αναπτύσσονται οι βάσεις δεδομένων, το API της εφαρμογής και οποιαδήποτε άλλη διαδικασία γίνεται και δεν είναι ορατή στους χρήστες, όπως το Third-Party Authentication, τα Push Notifications κλπ.

Mobile components

Στον τεχνικό τομέα του Mobile, προτεραιότητα αποτελεί η σχεδίαση του GUI *(Graphical User Interface)*, δηλαδή του περιβάλλοντος με το οποίο θα αλληλεπιδρούν οι χρήστες και στη συνέχεια, η σύνδεσή του με το API του Back-End.

Testing & Integration

Σε αυτή τη φάση της ανάπτυξης γίνεται η πλήρης ενσωμάτωση του Mobile τμήματος με όλες τις Back-End υπηρεσίες και στη συνέχεια επιλύνονται όποια σφάλματα προκύπτουν μέσω εκτεταμένων ελέγχων χρήσης της εφαρμογής.

Deployment & Maintenance (Containerization)

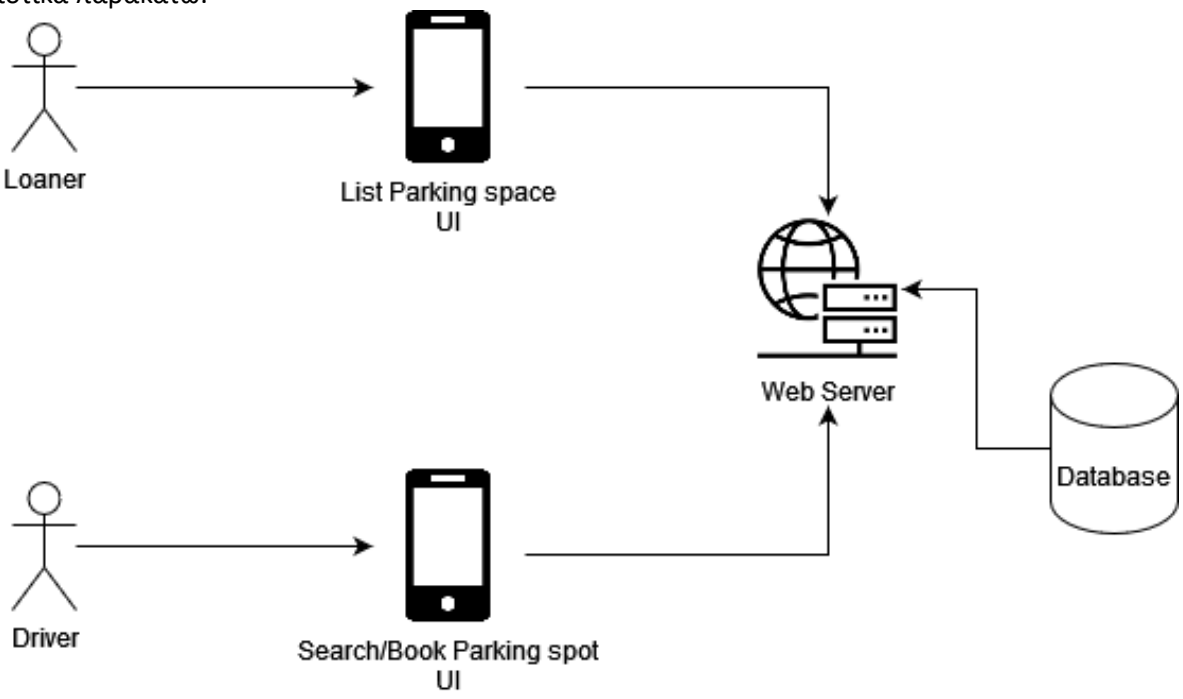
Σε αυτή τη φάση της ανάπτυξης γίνεται *Containerization* για όποια τμήματα της εφαρμογής χρειάζεται, έτσι ώστε να γίνει η προετοιμασία για τη δημοσίευση της εφαρμογής ενδεχομένως σε κάποιο Server. Ωστόσο, για τις ανάγκες της παρούσας εργασίας η δημοσίευση των components της εφαρμογής έγιναν μόνο σε τοπικό *(local)* επίπεδο.

ΚΕΦΑΛΑΙΟ 4°: ΑΡΧΙΤΕΚΤΟΝΙΚΗ ΚΑΙ ΤΕΧΝΟΛΟΓΙΕΣ

5.1 ΑΡΧΙΤΕΚΤΟΝΙΚΉ

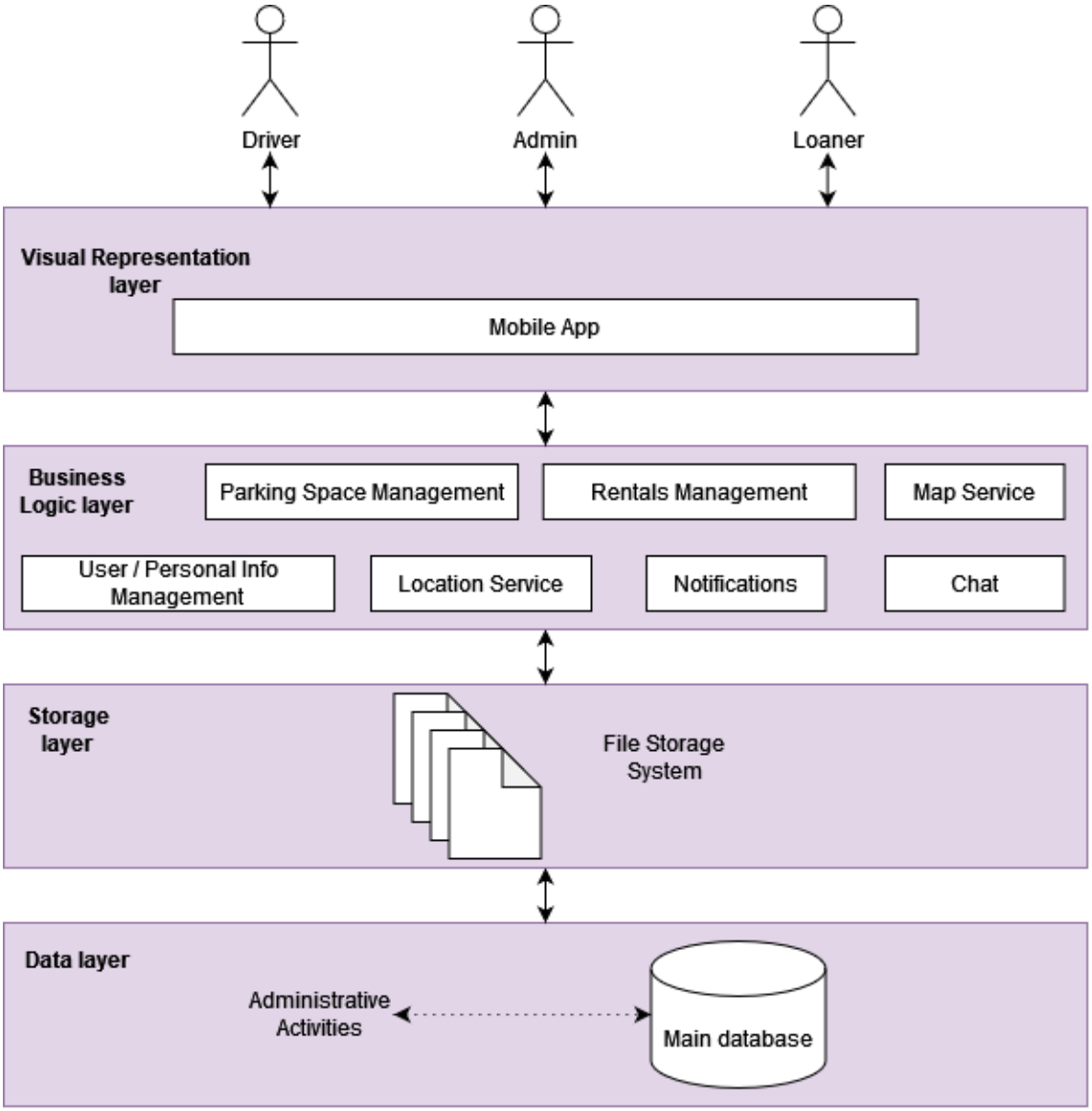
Η αρχιτεκτονική ενός λογισμικού ορίζεται από τη δομική οργάνωσή του. Ένα σύστημα αποτελείται από τμήματα (*components*), την αλληλεπίδραση μεταξύ αυτών (*interaction*), το περιβάλλον στο οποίο αυτά φιλοξενούνται (*hosting environment*) και τις σχεδιαστικές αρχές που εφαρμόζονται κατά την υλοποίησή τους (*principles of software design*). Ακόμη, σε κάποια συστήματα συμπεριλαμβάνεται η εξέλιξη (*evolution*) του λογισμικού σε μελλοντικό χρόνο.

Η αρχιτεκτονική ενός συστήματος δημιουργείται μετά την ενδελεχή ανάλυση απαιτήσεων και την ακριβή περιγραφή του λογισμικού. Για τις ανάγκες της συγκεκριμένης εργασίας, εφόσον ορίστηκε το *target group* των χρηστών σε συνδυασμό με τους ρόλους που μπορεί αυτοί να έχουν και οι λειτουργίες που θα εκτελεί η εφαρμογή, η αρχιτεκτονική δομή δίνεται ποιοτικά παρακάτω.



Εικόνα 10 Αρχιτεκτονική Συστήματος

Στη συνέχεια, έγινε έρευνα σχετικά με τα τεχνικά κομμάτια του λογισμικού και τις πιθανές τεχνολογίες ή και σχεδιαστικά πρότυπα που θα μπορούσαν να εφαρμοσθούν αποδοτικά στο σύστημα. Μια πιο λεπτομερής αρχιτεκτονική, συνεπώς, περιγράφεται με την εξής δομή: αρχικά, χρειάζεται μια βάση δεδομένων που θα εξυπηρετεί ως το κύριο μέσο αποθήκευσης και διαχείρισης πληροφοριών από τους διαχειριστές και τους απλούς χρήστες. Κατόπιν, είναι απαραίτητο να υπάρχει ένα σύστημα αποθήκευσης αρχείων, καθώς στην εφαρμογή θα υπάρχει η δυνατότητα προβολής και «ανεβάσματος» (*upload)* εικόνων. Στη συνέχεια, υλοποιείται η λογική της εφαρμογής, δηλαδή η διαχείριση θέσεων στάθμευσης από τους χρήστες, η διαχείριση των κρατήσεων, η ταυτοποίηση των χρηστών, η προβολή των θέσεων πάρκινγκ μέσω χαρτών, οι συνομιλίες μεταξύ χρηστών και οι ειδοποιήσεις. Τέλος, αναπτύσσεται το γραφικό περιβάλλον με το οποίο θα αλληλεπιδρούν οι χρήστες.



Εικόνα 11 Λεπτομερής Αρχιτεκτονική λογισμικού

5.2 ΤΕΧΝΟΛΟΓΙΕΣ

Για τις ανάγκες της συγκεκριμένης εργασίας τα τμήματα που συνθέτουν το παρόν σύστημα είναι τα εξής:

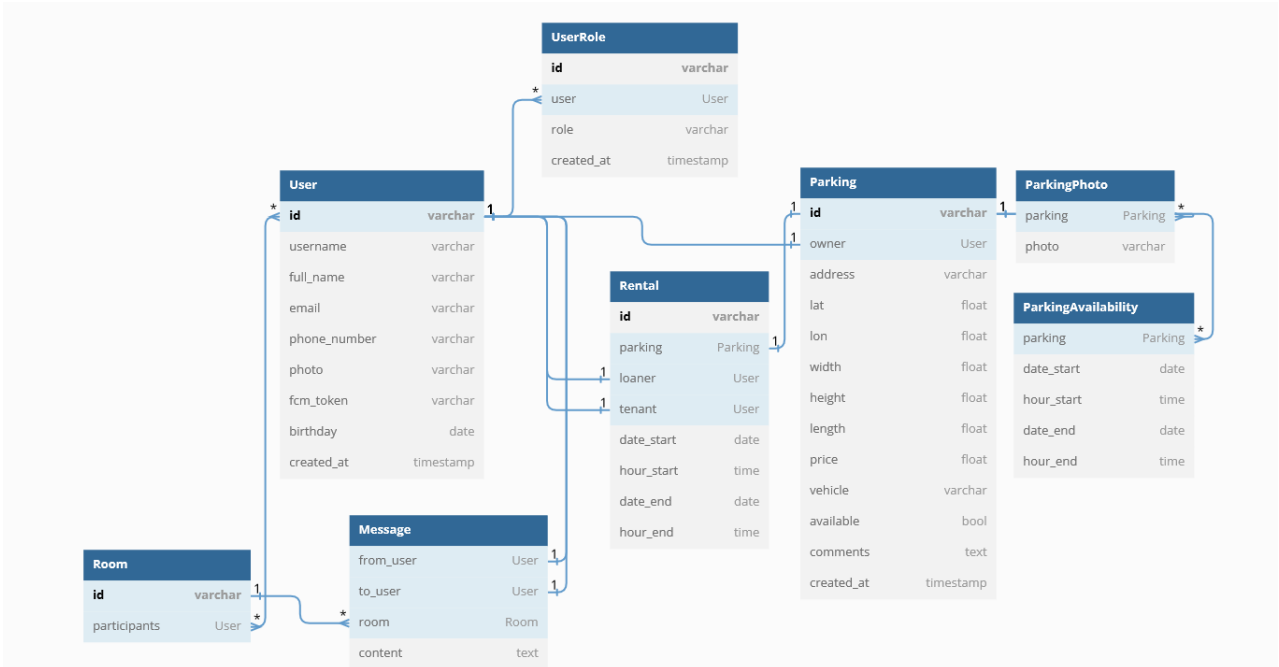
5.2.1 Backend component

Βάση δεδομένων

Αυτή είναι η κύρια βάση δεδομένων για την αποθήκευση structured δεδομένων. Εδώ συλλέγονται όλα τα στοιχεία των χρηστών, τα στοιχεία των θέσεων στάθμευσης, τα στοιχεία των ενοικιάσεων και οι συνομιλίες μεταξύ των χρηστών. Αυτή υλοποιήθηκε με PostgreSQL καθώς αποτελεί μια βάση δεδομένων που μπορεί να είναι απομακρυσμένη και συνδέεται με το API μέσω του pip package psycopg2-binary.

Εικόνα 12 Λογότυπο PostgreSQL

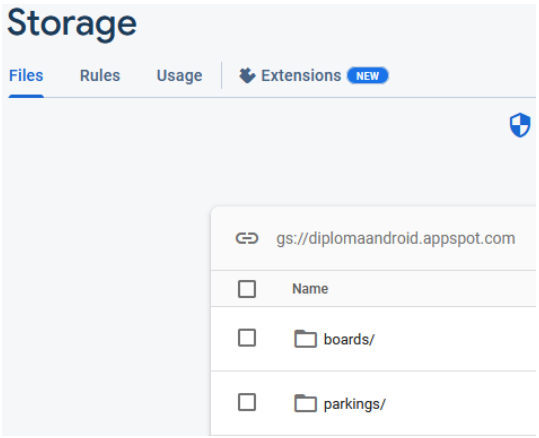
Η δομή της βάσης της εφαρμογής υλοποιήθηκε με το εξής σχήμα:



Εικόνα 13 Δομή κύριας βάσης δεδομένων

Αποθηκευτικός χώρος αρχείων

Για αυτή τη δομή δεδομένων χρειάζεται η χρήση ενός file system για την εύκολη αποθήκευση unstructured αρχείων, όπως οι φωτογραφίες των θέσεων πάρκινγκ και οι φωτογραφίες των προφίλ των χρηστών. Για το λόγο αυτό χρησιμοποιήθηκε το Firebase Storage με το εξής σχήμα:



Εικόνα 14 Firebase Storage δομή αρχείων

API

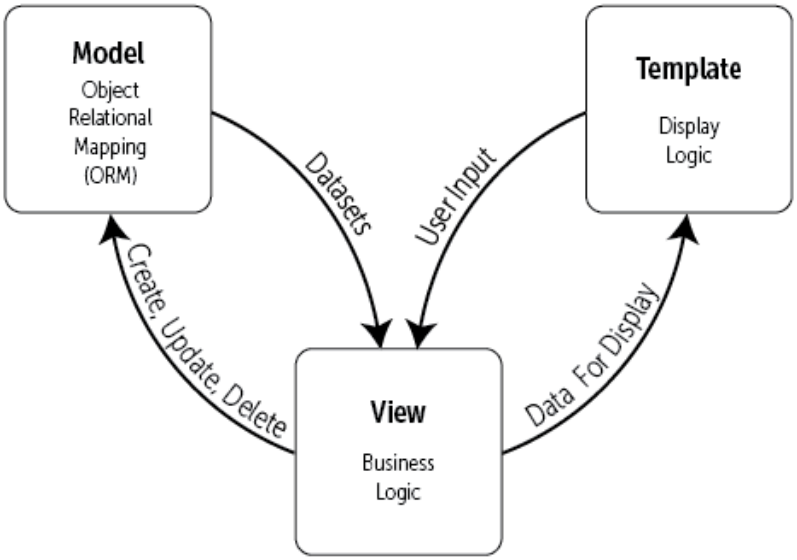
Για την επικοινωνία της mobile εφαρμογής με τις λειτουργίες της βάσης, της ταυτοποίησης των χρηστών με ή χωρίς third-party παραγόντων και των manual ειδοποιήσεων των χρηστών χρειάζεται η ανάπτυξη ενός API. Για τη συγκεκριμένη εργασία το API υλοποιήθηκε με Python σε Django Rest Framework.

Εικόνα 15 Λογότυπο Django Rest Framework

MVT πρότυπο σχεδίασης

Για την υλοποίηση του Backend χρησιμοποιήθηκε το Django Rest Framework, το οποίο εφαρμόζει το design pattern Model View Template (MVT). Το πρότυπο αυτό αποτελεί μια προσέγγιση του κλασικού πρότυπου Model View Controller, με τη διαφορά ότι τα Templates

αντιστοιχούν στα Views και τα Views στους Controllers. Στο πρότυπο σχεδίασης MVT, το Django framework διαχειρίζεται μόνο του το μέρος του Controller.



Εικόνα 16 MVT πρότυπο σχεδίασης

Ασύγχρονο κανάλι για συνομιλίες

Το κανάλι αυτό είναι υπεύθυνο για τα chat μεταξύ των χρηστών, τα οποία μπορούν να γίνονται παράλληλα μεταξύ πολλαπλών χρηστών και κυρίως ασύγχρονα. Για την εφαρμογή χρησιμοποιήθηκε το Uvicorn, το οποίο αποτελεί ένα low-level interface για την ασύγχρονη επικοινωνία μεταξύ server και εφαρμογής.

Εικόνα 17 Λογότυπο Uvicorn

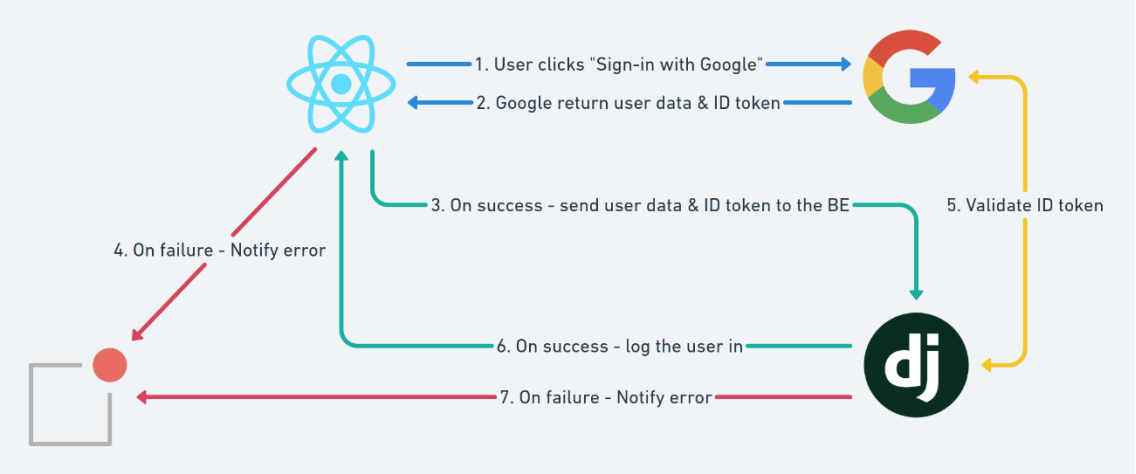
Λειτουργεί ως ένας ασύγχρονος web server δηλαδή ένας *ASGI (async server gateway interface)* και συμβάλλει στην εκτέλεση πολλαπλών αιτημάτων στα API που εκθέτει από πολλαπλούς χρήστες σε ασύγχρονο περιβάλλον καθώς λειτουργεί και με Web Sockets, τα οποία χρησιμοποιήθηκαν στο Mobile τμήμα της εφαρμογής.

```
(venv) PS D:\uni\MSc\diploma\RentMyParkingSpace> uvicorn app.asgi:application --reload --host 0.0.0.0
INFO:     Will watch for changes in these directories: ['D:\\uni\\MSc\\diploma\\RentMyParkingSpace']
INFO:     Uvicorn running on http://0.0.0.0:8000 (Press CTRL+C to quit)
INFO:     Started reloader process [8112] using WatchFiles
INFO:     Started server process [13840]
INFO:     Waiting for application startup.
INFO:     ASGI 'lifespan' protocol appears unsupported.
INFO:     Application startup complete.
```

Εικόνα 18 Παράδειγμα εκτέλεσης ASGI server και API

Google Authentication

Οι χρήστες έχουν τη δυνατότητα να εγγραφούν στο σύστημα μέσω Username και Password ή μέσω του Google λογαριασμού τους. Η δεύτερη επιλογή επιτυγχάνεται μέσω του *social_django package.* Οι χρήστες συνδέονται με το Google λογαριασμό τους και το σύστημα λαμβάνει ένα ID token και τις βασικές πληροφορίες του λογαριασμού. Το ID token επικυρώνεται και στη συνέχεια εάν είναι έγκυρο, στο backend του συστήματος παράγονται τα access tokens (JWT) και αποθηκεύεται ο νέος χρήστης βάση.



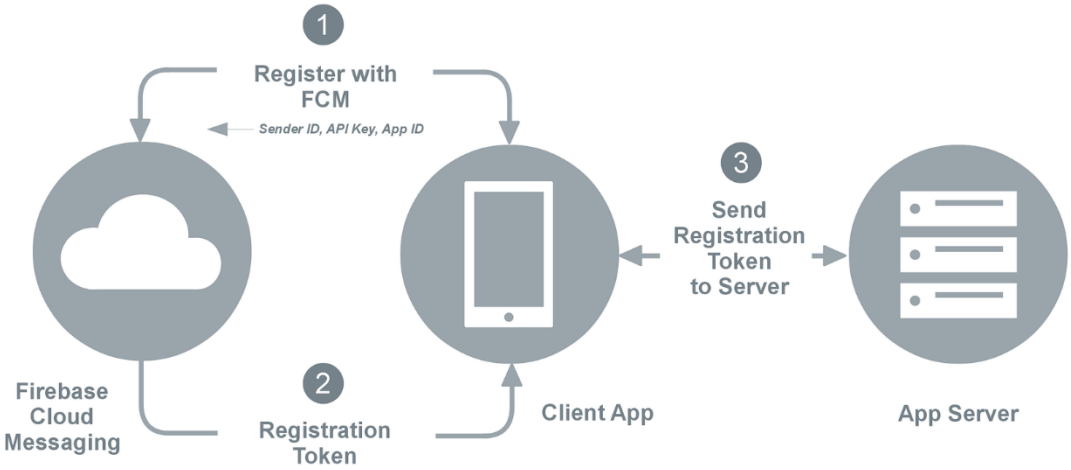
Εικόνα 19 Δομή σύνδεσης μέσω Google

Cloud Messaging component

Το τμήμα αυτό είναι υπεύθυνο για τα push notifications προς τους διάφορους χρήστες. Οι ειδοποιήσεις αυτές έγιναν μέσω του *FCM (Firebase Cloud Messaging)*.

Εικόνα 20 Λογότυπο Firebase Cloud Messaging

Για την αποστολή των ειδοποιήσεων αυτών αποθηκεύεται στη βάση το FCM token του κάθε χρήστη και για την εφαρμογή τους στο *DRF (Django Rest Framework)* χρησιμοποιήθηκε *το pip package pyfcm*. Αντίστοιχα, για την εφαρμογή τους στο Android χρησιμοποιήθηκε η βιβλιοθήκη *com.google.firebase.messaging* και η κλάση *FirebaseMessagingService.*



Εικόνα 21 Δομή σύνδεσης εφαρμογής, server με το FCM

Containerization (Docker)

Το Docker είναι μια πρακτική φιλοξενίας ολοκληρωμένων συστημάτων σε κατάσταση παραγωγής ή μη σε εξατομικευμένα κοντεινερ, δηλαδή σε εικονικά περιβάλλοντα στα οποία εγκαθίστανται. Χρησιμοποιήθηκε συγκεκριμένα για την ανάρτηση και εγκατάσταση του back end και της βάσης δεδομένων της εφαρμογής σε τοπικό δίκτυο και άρα αφορά τη φάση του deployment και maintenance στην ανάπτυξη του λογισμικού**.**

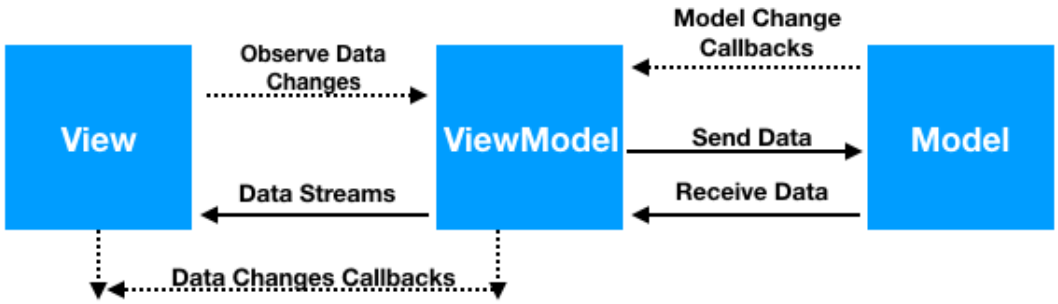
5.2.2 Mobile component

Γλώσσα προγραμματισμού Kotlin

Το mobile τμήμα της εφαρμογής υλοποιήθηκε στο περιβάλλον του Android Studio με τη γλώσσα Kotlin, η οποία αποτελεί μια αντικειμενοστραφής γλώσσα προγραμματισμού που σχεδιάστηκε από την ομάδα JetBrains. Η Kotlin τρέχει πάνω στην Εικονική μηχανή της Java.

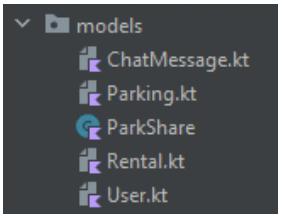
MVVM πρότυπο σχεδίασης

Το MVVM design pattern είναι ένα αρχιτεκτονικό πρότυπο οργάνωσης κώδικα το οποίο διαχωρίζει ό,τι αφορά στην παρουσίαση (UI) και την αλληλεπίδραση με το χρήστη από τη βασική λογική του λογισμικού (business logic). Βάσει του προτύπου αυτού, ο κώδικας μιας εφαρμογής χωρίζεται σε τρία επίπεδα; Τα Models, τα Views και τα ViewModels.



Εικόνα 22 Πρότυπο αρχιτεκτονικής MVVM

1. Models
   Εδώ ορίζονται οι δομές των δεδομένων για κάθε τύπο πληροφορίας που θα παρουσιάζει η mobile εφαρμογή (Χρήστης, Θέση Παρκινγκ, Ενοικίαση, Μήνυμα).



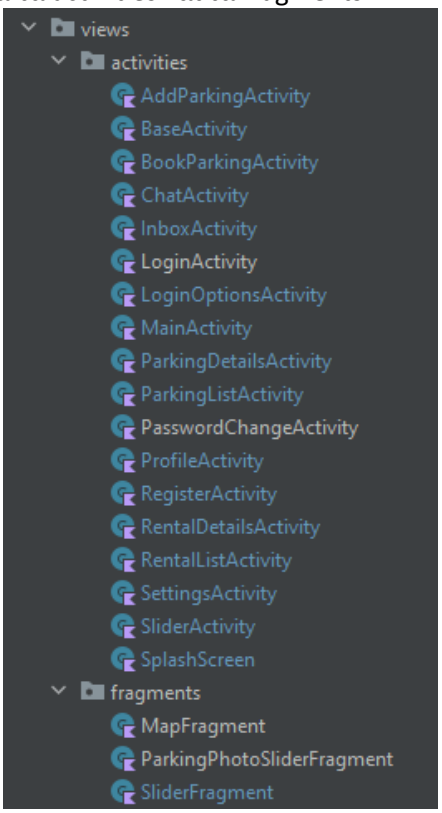
Εικόνα 23 Models

```
Angelica Theodorou
@Parcelize
data class UserRole (
    @SerializedName("role") internal val role: String? = null,
) : Parcelable
Angelica Theodorou
@Parcelize
data class User(
    @SerializedName("id") internal val id: UUID? = null,
    @SerializedName("google_ID") internal val google_ID: String? = null,
    @SerializedName("id_token") internal val id_token: String? = null,
    @SerializedName("fcm_token") internal var fcm_token: String? = null,
    @SerializedName("username") internal var username: String? = null,
    @SerializedName("password") internal var password: String? = null,
    @SerializedName("full_name") internal var full_name: String? = null,
    @SerializedName("email") internal var email: String? = null,
    @SerializedName("phoneNumber") internal var phoneNumber: String? = null,
    @SerializedName("photo") internal var photo: String? = null,
    @SerializedName("birthday") internal var birthday: String? = null,
    @SerializedName("roles") internal var roles: ArrayList<UserRole>,
): Parcelable

Angelica Theodorou
data class AuthResponse (
    @SerializedName("refresh")
    val refresh: String?,
    @SerializedName("access")
    val access: String? = null,
    @SerializedName("user")
    val user: User? = null
)
```

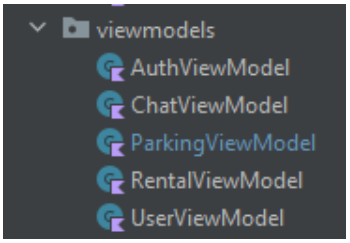
Εικόνα 24 User Model

2. Views
   Εδώ ορίζονται οι οθόνες που θα βλέπει ο χρήστης όσο χρησιμοποιεί την εφαρμογή και εν προκειμένω είναι όλα τα activities και τα fragments.



Εικόνα 25 Views

3. ViewModels
   Τα ViewModels αποτελούν τις δομές οι οποίες λαμβάνουν τα δεδομένα που χρησιμοποιούνται από κάθε View από το API.

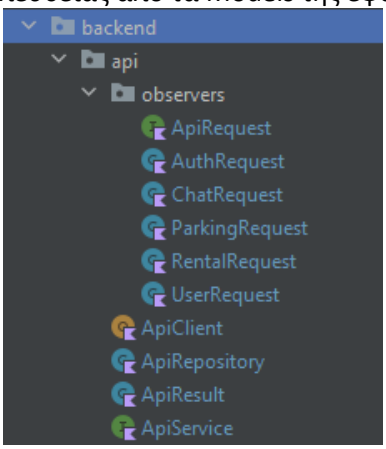


Εικόνα 26 ViewModels

***Retrofit*: API requests component**

Η Retrofit αποτελεί μια βιβλιοθήκη σχεδιασμένη για την εύκολη διαχείριση RESTful υπηρεσιών. Για τις ανάγκες της εργασίας χρησιμοποιήθηκε στη διαχείριση του κεντρικού API του Backend. Η βιβλιοθήκη αυτή συνδυάζεται εύκολα και αποδοτικά με το MVVM design

pattern, καθώς χρησιμοποιεί μοντέλα αντικειμένων για το serialization των responses από και σε JSON, τα οποία μπορεί να λαμβάνει απευθείας από τα Models της εφαρμογής.



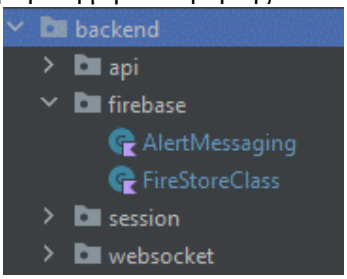
Εικόνα 27 Τμήμα κώδικα API requests

***Firebase Storage*: Διαχείριση Αποθηκευτικού χώρου αρχείων**

Για τη διαχείριση των αρχείων χρησιμοποιήθηκε η βιβλιοθήκη της Firebase Storage. Συγκεκριμένα έγινε χρήση της για το ανέβασμα εικόνων στην εφαρμογή και για τη διαγραφή τους, όποτε γίνεται διαγραφή λογαριασμού ή διαγραφή ενός εγγεγραμμένου χώρου πάρκινγκ στο σύστημα.

***Firebase Cloud Messaging*: Διαχείριση ασύγχρονων ειδοποιήσεων**

Για τη διαχείριση των ειδοποιήσεων μέσω Push Notifications είτε από μηνύματα είτε από ενημερώσεις για ενοικιάσεις, χρησιμοποιήθηκε η βιβλιοθήκη της Firebase Cloud Messaging.

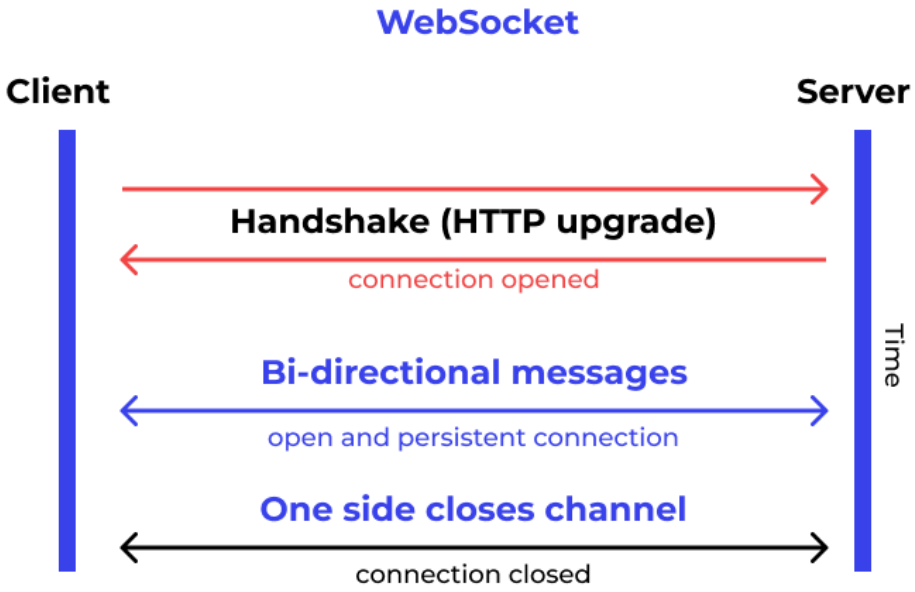


Εικόνα 28 Τμήμα κώδικα Firebase

Λοιπές λειτουργίες

1. Sessions
   Στα sessions αποθηκεύονται διάφορες πληροφορίες που αφορούν κάθε χρήστη όπως τα access tokens (JWT), το fcm (firebase cloud messaging) token της συσκευής του, οι προτιμήσεις του στη γλώσσα και τα φίλτρα κλπ. Αυτές οι πληροφορίες αφορούν αποκλειστικά κάθε χρήστη και ενδεχομένως κάθε φορά που χρησιμοποιεί την εφαρμογή αυτές αλλάζουν και άρα αποθηκεύονται προσωρινά στα Shared Preferences της εφαρμογής.
2. Web Sockets

Τα web sockets είναι απαραίτητα για την επικοινωνία με τον ασύγχρονο server του chat και χρησιμοποιήθηκαν για την αποστολή και λήψη μηνυμάτων στις συνομιλίες μεταξύ των χρηστών.



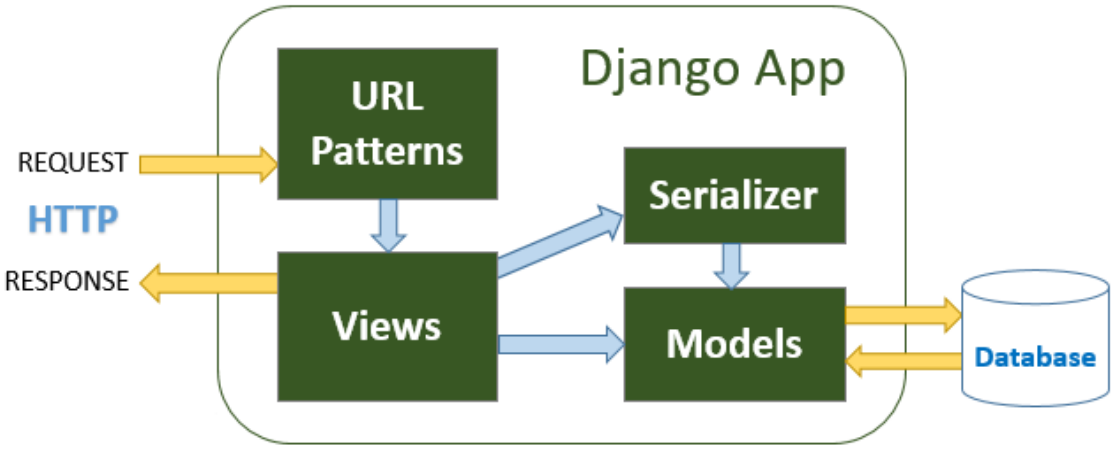
Εικόνα 29 Λειτουργία Web Socket

3. Google Authentication
   Για την ταυτοποίηση χρηστών μέσω Google χρησιμοποιήθηκε η βιβλιοθήκη της GoogleSignIn και μέσω αυτής το σύστημα λαμβάνει τις βασικές πληροφορίες των λογαριασμών Google και τις συγχρονίζει με τους λογαριασμούς στο Backend της εφαρμογής.
4. Βοηθητικές κλάσεις που ήταν απαραίτητες για validation των δεδομένων που δίνουν οι χρήστες και των δεδομένων που λαμβάνει το σύστημα από το Backend, για την παρουσίαση δεδομένων στο UI, για την ανάκτηση διευθύνσεων και τοποθεσιών μέσω location services κλπ.

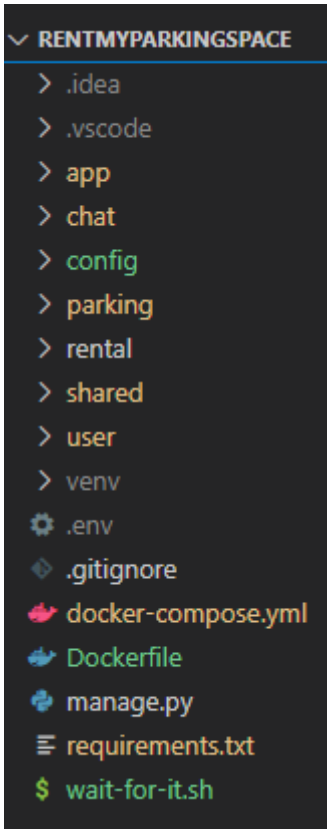
ΚΕΦΑΛΑΙΟ 5°: ΥΛΟΠΟΙΗΣΗ ΕΦΑΡΜΟΓΗ

6.1 ΔΟΜΉ ΕΦΑΡΜΟΓΉΣ

6.1.1 Back-End



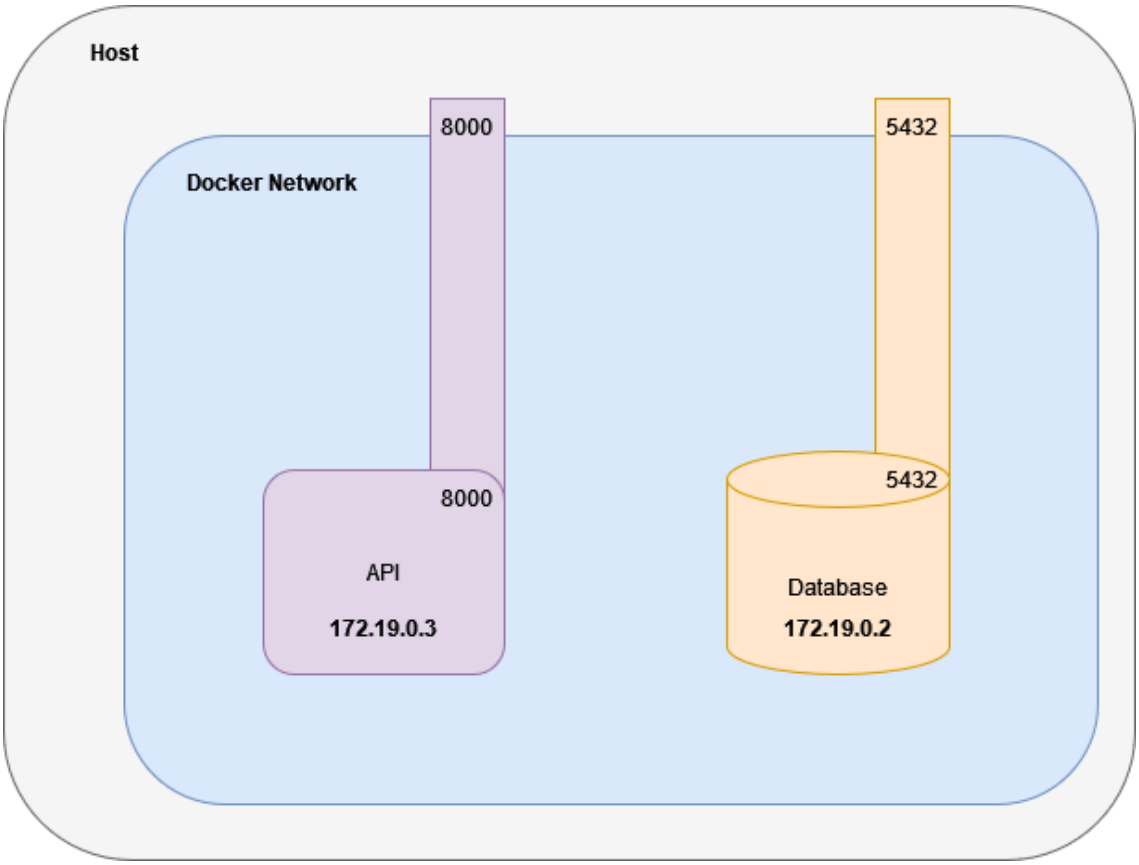
Εικόνα 30 Διάγραμμα σύνδεσης Django Rest Framework



Εικόνα 31 Δομή εφαρμογής στο Visual Code

Το API της εφαρμογής σε συνδυασμό με την κύρια βάση δεδομένων είναι containerized και εκτελούνται μέσω του docker-compose.yml αρχείου.

Παρακάτω παρουσιάζεται ένα διάγραμμα για την ευκολότερη κατανόηση της χρήσης και αλληλεξάρτησης μεταξύ του API και της βάσης δεδομένων μέσω του Docker στην εφαρμογή.



Εικόνα 32 Διάγραμμα δικτύου API και βάσης δεδομένων

Ο server της εφαρμογής ενεργοποιείται μέσω του uvicorn framework, το οποίο εκκινεί ένα ασύγχρονο κανάλι που συνδέεται με το server του Django Rest Framework.

Στο φάκελο “config” και στο αρχείο “.env” βρίσκονται όλα τα client secrets και τα api keys που είναι απαραίτητα για τη σύνδεση του συστήματος με το Google και τη Firebase.

Στο φάκελο “app” εμπεριέχονται όλα τα αρχεία που αφορούν στις ρυθμίσεις, το Routing, τη διαχείριση των γενικών URL και των σύγχρονων και ασύγχρονων καναλιών της εφαρμογής.

Γενικά, τα πρότζεκτ του Django Rest Framework χωρίζουν τις υπηρεσίες τους σε εφαρμογές “apps”. Κάθε εφαρμογή ορίζει τα μοντέλα της σε συνδυασμό με τα migrations τους στη βάση, τα views της, τα URLs της και ό,τι άλλο αφορά στο εκάστοτε “app”, όπως serializers, φίλτρα, templates κ.ο.κ.

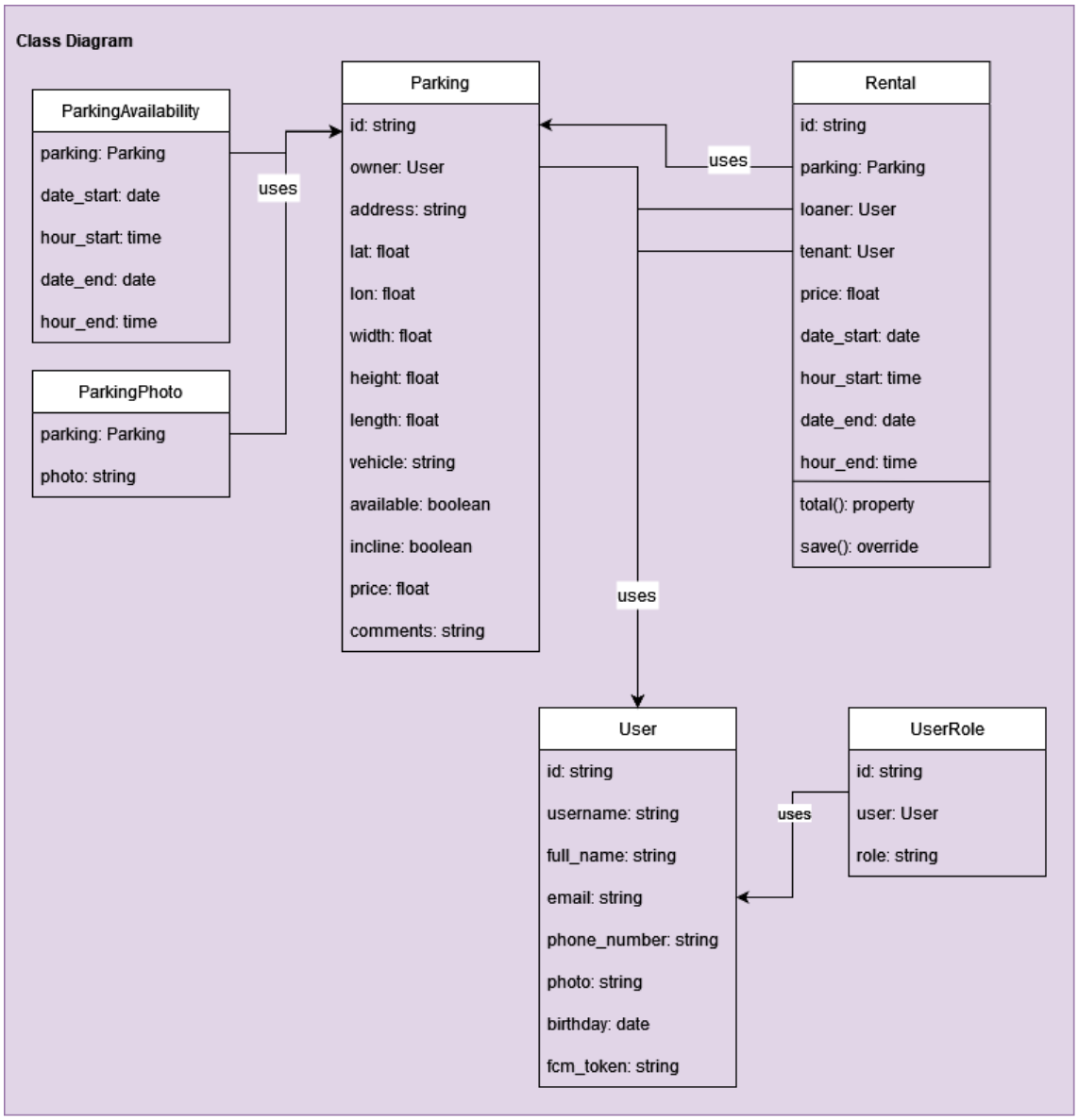
Το “chat” αποτελεί μια εφαρμογή στο DRF και άρα έχει δικό του φάκελο. Στο φάκελο αυτόν βρίσκεται ο ορισμός του μοντέλου μιας συνομιλίας (μήνυμα – δωμάτιο) όπως αυτό περιγράφεται στην εικόνα 13 και τα migrations του μοντέλου αυτού στη βάση. Ακόμη ορίζονται τα requests του API που αφορούν στο chat καθώς και η λογική της αποστολής και παραλαβής ασύγχρονων μηνυμάτων σε μια συνομιλία από τους χρήστες στο σύστημα. Τέλος, σε αυτή την εφαρμογή ορίζονται και οι ειδοποιήσεις που λαμβάνουν οι χρήστες σε μορφή push notification μέσω του FCM.

Αντίστοιχα, το “parking” αποτελεί μια εφαρμογή στο DRF και άρα έχει δικό του φάκελο. Στο φάκελο αυτόν ορίζονται τα μοντέλα που αφορούν στους χώρους στάθμευσης, δηλαδή μια θέση παρκινγκ, η διαθεσιμότητα της θέσης και οι εικόνες της. Τέλος, ορίζονται τα requests του API που αφορούν στα πάρκινγκ.

Το “rental” αποτελεί μια εφαρμογή στο DRF και άρα έχει δικό του φάκελο. Σε αντιστοιχία με την παραπάνω εφαρμογή σε αυτόν τον φάκελο ορίζεται το μοντέλο της κράτησης και υλοποιούνται τα requests του API που αφορούν στις ενοικιάσεις.

Στο φάκελο “shared” βρίσκονται οι υπόλοιπες υπηρεσίες που απαιτούνται για την ομαλή λειτουργία του API, όπως τα permissions ανάλογα με το ρόλο των χρηστών, οι validators των δεδομένων που έρχονται από το Mobile κ.ο.κ.

Τέλος, στο φάκελο “user” ορίζεται η ομώνυμη εφαρμογή του DRF. Εδώ, υλοποιείται το μοντέλο των χρηστών, η ταυτοποίηση αυτών μέσω Google ή μέσω Username-Password και τα requests του API που αφορούν στη σύνδεση, εγγραφή και διαχείριση του προφίλ των χρηστών. Δίνεται παρακάτω το διάγραμμα κλάσεων του API για ευκολότερη κατανόηση της δομής του.

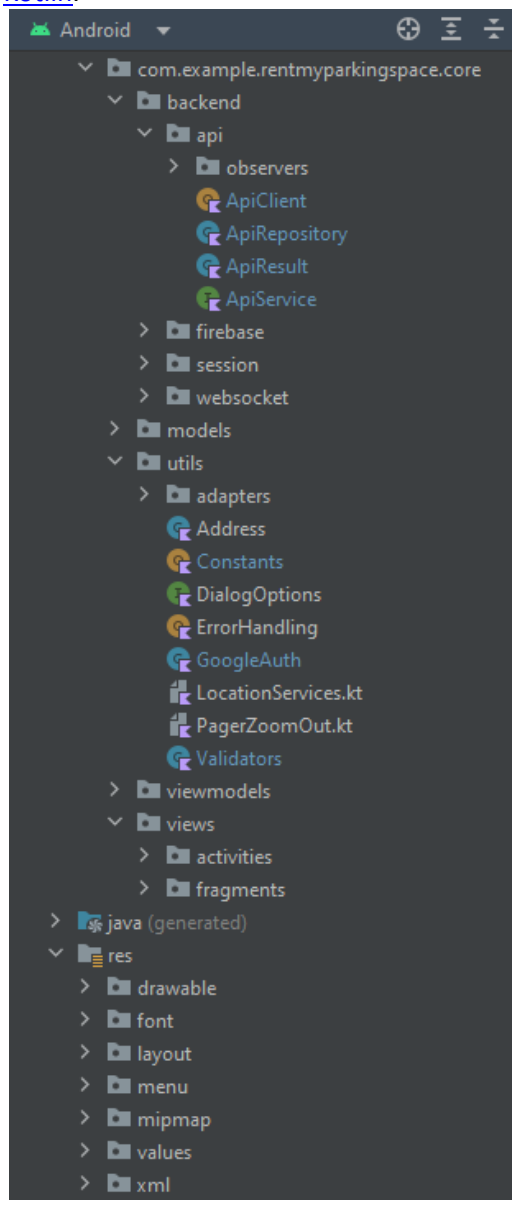


Εικόνα 33 Διάγραμμα κλάσεων API

Τα υπόλοιπα αρχεία που φαίνονται στη δομή του API αφορούν στα απαραίτητα πακέτα που εγκαθίστανται για τη λειτουργία του (requirements.txt) και το αρχείο “wait-for-it.sh” αποτελεί ένα shell script που συμβάλλει στη λειτουργία του API και της βάσης δεδομένων σε σωστή σειρά όταν εκτελούνται από το Docker.

6.1.2 Mobile

Για την υλοποίηση του Mobile χρησιμοποιήθηκε το περιβάλλον του Android Studio με τη γλώσσα προγραμματισμού Kotlin.



Εικόνα 34 Δομή mobile στο Android Studio

Στο σύστημα εφαρμόστηκε το πρότυπο σχεδίασης MVVM όπως αναφέρεται παραπάνω και συνεπώς η δομή της εφαρμογής ακολουθεί αυτό το πρότυπο δομικά στο μεγαλύτερο μέρος της. Στους φακέλους “models”, “viewmodels” και “views” εμπεριέχονται τα ομώνυμα τμήματα του συστήματος, δηλαδή τα μοντέλα, οι συνδέσεις των μοντέλων με τα τμήματα του UI και το Backend και τα τμήματα του UI, αντίστοιχα.

Στο φάκελο “backend” ορίζονται όλες οι λειτουργίες που αφορούν στο API, τις υπηρεσίες της Firebase, τα προσωπικά Sessions των χρηστών και τα WebSockets για τις συνομιλίες των χρηστών.

Στο φάκελο “utils” βρίσκονται οι βοηθητικές λειτουργίες της εφαρμογής που είναι απαραίτητες για το γραφικό περιβάλλον που αλληλεπιδρούν οι χρήστες, όπως διάφοροι Adapters και Sliders, αλλά και για τις υπηρεσίες Backend, όπως λειτουργίες για την ταυτοποίηση μέσω Google, για την εύρεση των διευθύνσεων των χρηστών μέσω της υπηρεσίας του GPS κ.ο.κ.

Τέλος, στο φάκελο “res” περιέχονται τα resources της εφαρμογής, δηλαδή όλα τα στατικά αρχεία του UI.

ΚΕΦΑΛΑΙΟ 6°: ΧΡΗΣΗ ΕΦΑΡΜΟΓΗΣ

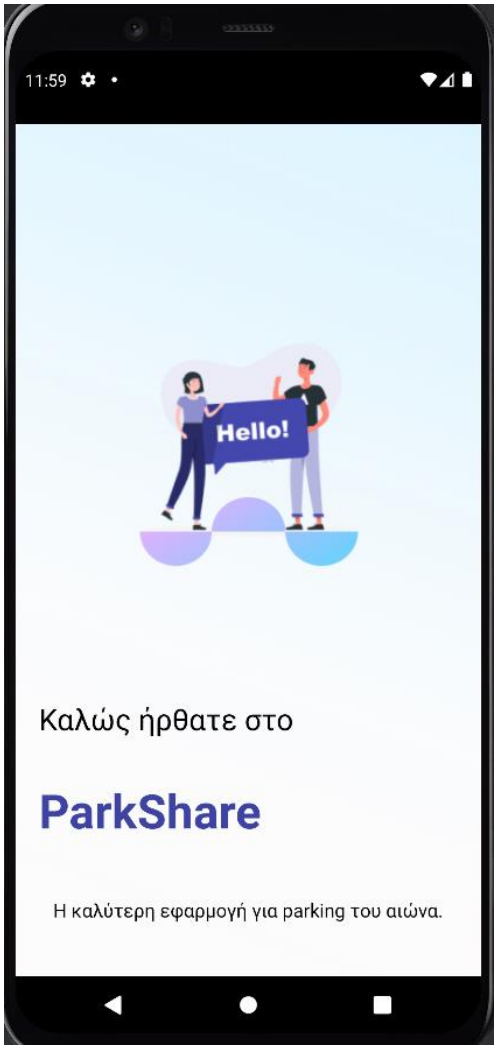
7.1 ΕΓΚΑΤΑΣΤΑΣΗ

Για τη σωστή εκτέλεση της εφαρμογής σε τοπικό επίπεδο χρειάζεται η χρήση του Docker. Καθώς έχει γίνει containerization σε όλο το backend κομμάτι του συστήματος για την εύκολη εγκατάσταση και χρήση του, είναι απαραίτητο αρχικά να εκτελεστεί και να λειτουργήσει σωστά η βάση δεδομένων, το API και ο ασύγχρονος server.

Εικόνα 35 Εκτέλεση βάσης, API και ασύγχρονου server

Σε δεύτερο στάδιο, είναι απαραίτητο να γίνει εγκατάσταση της εφαρμογής σε μια συσκευή Android, η οποία να υποστηρίζει Google Services. Στα παρακάτω παραδείγματα γίνεται χρήση του emulator Android Studio.

Εικόνα 36 Εφαρμογή ParkShare σε Android Emulator παράδειγμα 1



Εικόνα 37 Εφαρμογή ParkShare σε Android Emulator παράδειγμα 2

7.2 ΕΓΧΕΙΡΙΔΙΟ ΧΡΗΣΤΗ

7.2.1 Σύνδεση/Εγγραφή

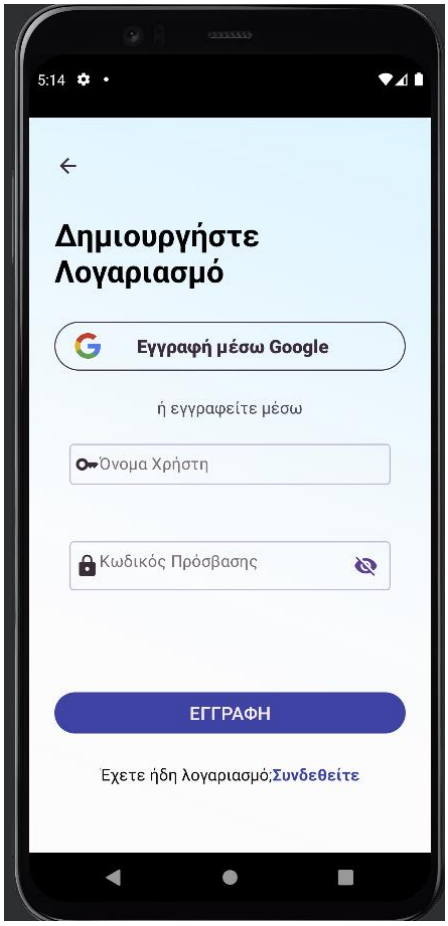
Αφού γίνει επιτυχής εγκατάσταση της εφαρμογής στην Android συσκευή και δοθούν οι απαραίτητες άδειες, ο χρήστης είναι έτοιμος να συνδεθεί με το λογαριασμό του ή να δημιουργήσει νέο στο σύστημα.

Υπάρχουν τρείς επιλογές για τη σύνδεση: ο χρήστης μπορεί να συνδεθεί μέσω Google, μέσω Username και Password ή να εγγραφεί εάν δεν έχει ήδη λογαριασμό.

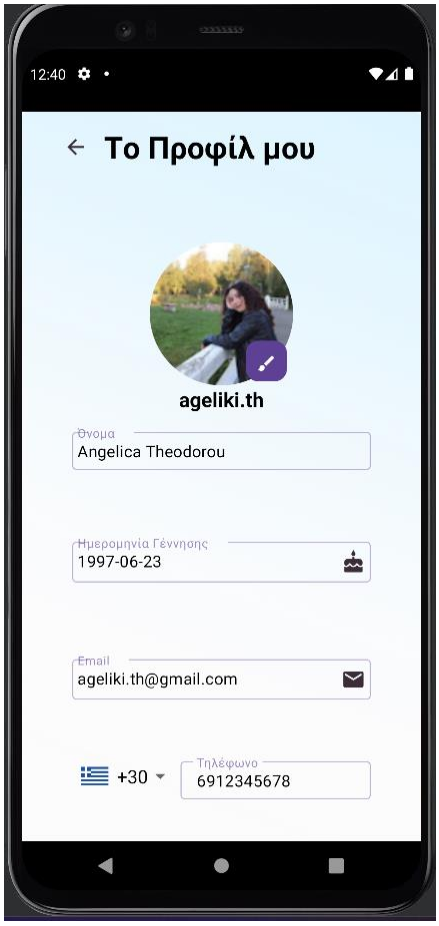
Εικόνα 38 ParkShare: Εισαγωγική οθόνη

Εικόνα 39 ParkShare: Οθόνη σύνδεσης

Σε περίπτωση που ο χρήστης δεν έχει λογαριασμό, εγγράφεται στο σύστημα αντίστοιχα μέσω Google ή μέσω Username και Password. Στη συνέχεια συμπληρώνει τα υπόλοιπα απαραίτητα στοιχεία του.



Εικόνα 40 ParkShare: Οθόνη Εγγραφής

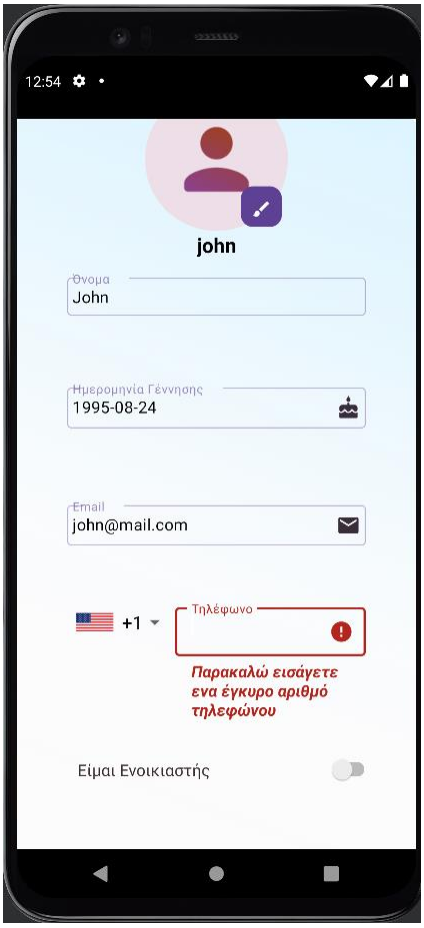


Εικόνα 41 ParkShare: Συμπλήρωση στοιχείων προφίλ

Ενδεικτικές οθόνες με μηνύματα σφαλμάτων στις φόρμες εγγραφής και σύνδεσης.

Εικόνα 42 ParkShare: Οθόνη σύνδεσης, σφάλμα

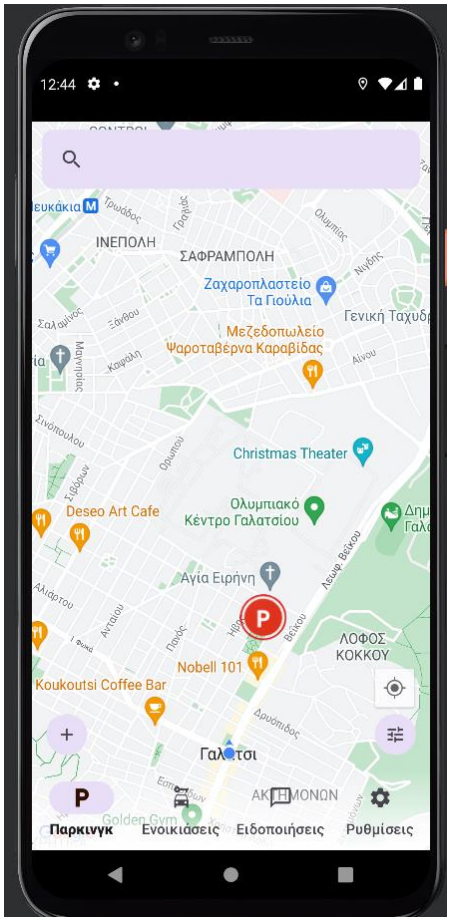
Εικόνα 43 ParkShare: Οθόνη συμπλήρωσης προφίλ, σφάλμα 1



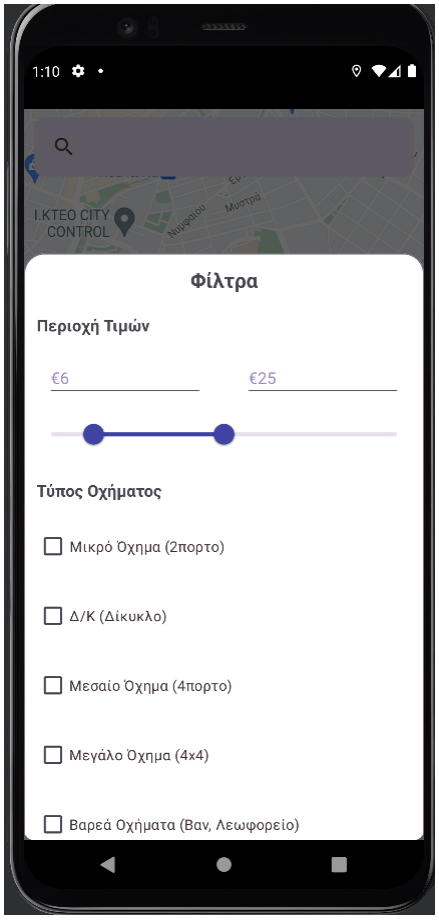
Εικόνα 44 ParkShare: Οθόνη συμπλήρωσης προφίλ, σφάλμα 2

7.2.2 Χάρτης/Αναζήτηση θέσης στάθμευσης

Αφού ο χρήστης συνδεθεί στο λογαριασμό του, μπορεί να περιηγηθεί στην εφαρμογή και να αναζητήσει θέσεις στάθμευσής, βάσει περιοχής, τιμών, είδους οχήματος, κλπ. Ακόμη, έχει τη δυνατότητα να καταγράψει το δικό του χώρο παρκινγκ εφόσον είναι ενοικιαστής, να ελέγξει τις ενοικιάσεις του και να ρυθμίσει το προφίλ του.



Εικόνα 45 ParkShare: Βασική οθόνη, Χάρτης

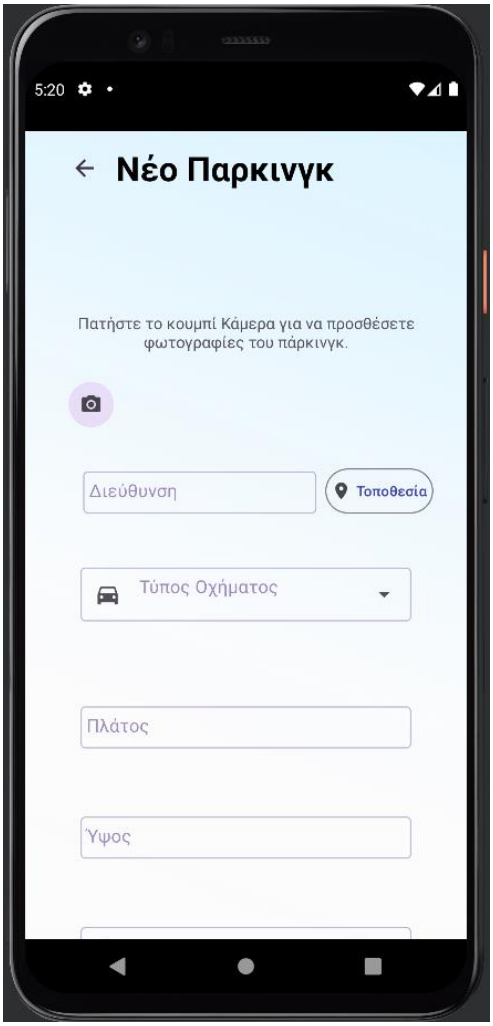


Εικόνα 46 ParkShare: Βασική οθόνη, Φίλτρα

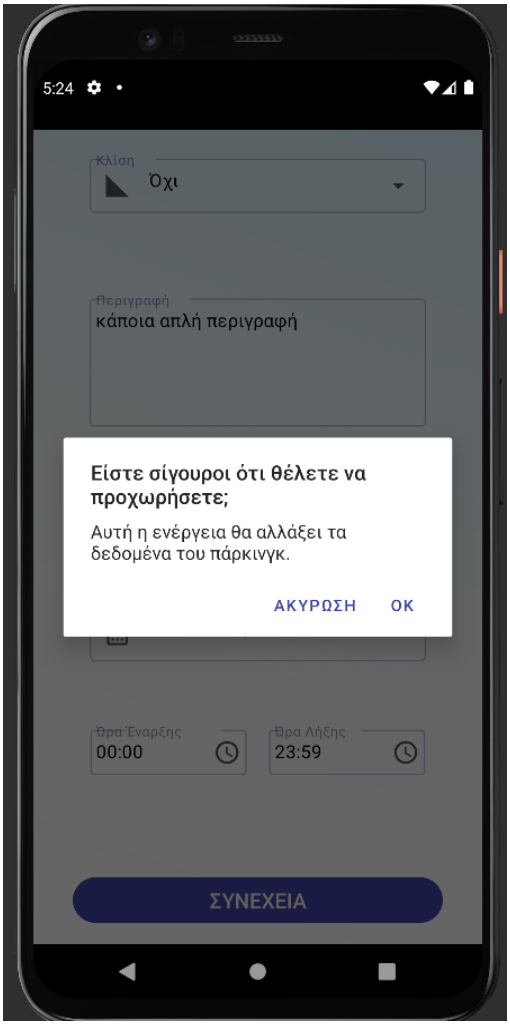
Εικόνα 47 ParkShare: Βασική οθόνη, Αναζήτηση Παρκινγκ

7.2.3 Προσθήκη νέας θέσης στάθμευσης

Κάθε χρήστης εφόσον έχει ρόλο ενοικιαστή μπορεί να καταγράψει τον ιδιωτικό χώρο πάρκινγκ του στην εφαρμογή πατώντας το «+» κάτω αριστερά στην οθόνη του χάρτη (βλ. Εικόνα 26).



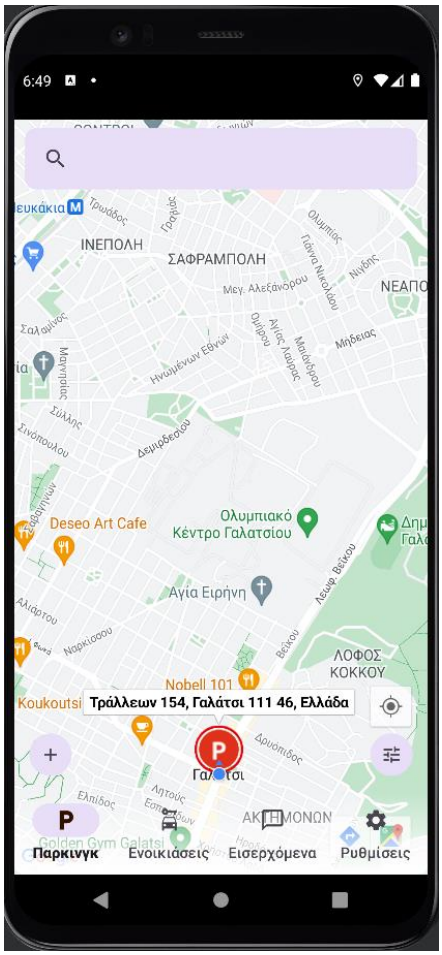
Εικόνα 48 ParkShare: Προσθήκη νέας θέσης πάρκινγκ, παράδειγμα 1



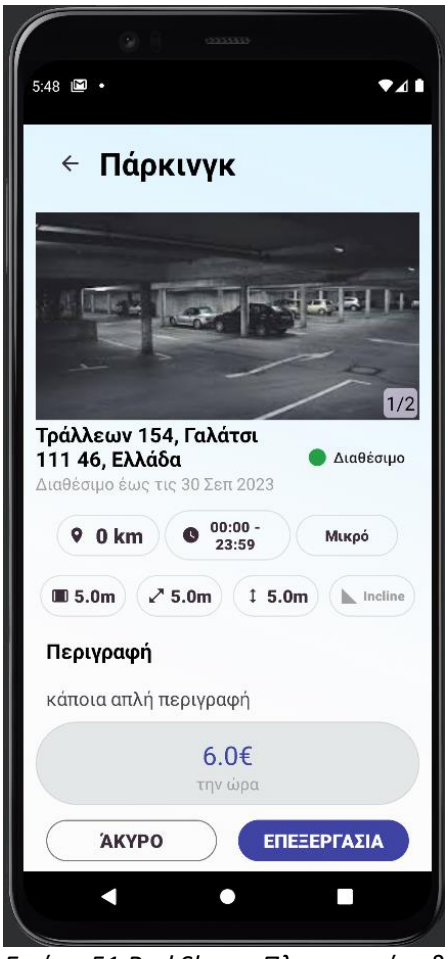
Εικόνα 49 ParkShare: Προσθήκη νέας θέσης πάρκινγκ, παράδειγμα 2

7.2.4 Προβολή πληροφοριών θέσης στάθμευσης

Μετά την προσθήκη νέων χώρων παρκινγκ κάθε χρήστης μπορεί να δει τις πληροφορίες που αφορούν σε κάθε πάρκινγκ, από τις ώρες διαθεσιμότητας έως και την ακριβή διεύθυνση, την απόσταση από την εκάστοτε τοποθεσία του χρήστη, τη μέση τιμή ανά ώρα, κάποιες ενδεικτικές φωτογραφίες του χώρου, κ.ο.κ.



Εικόνα 50 ParkShare: Θέση πάρκινγκ στο χάρτη

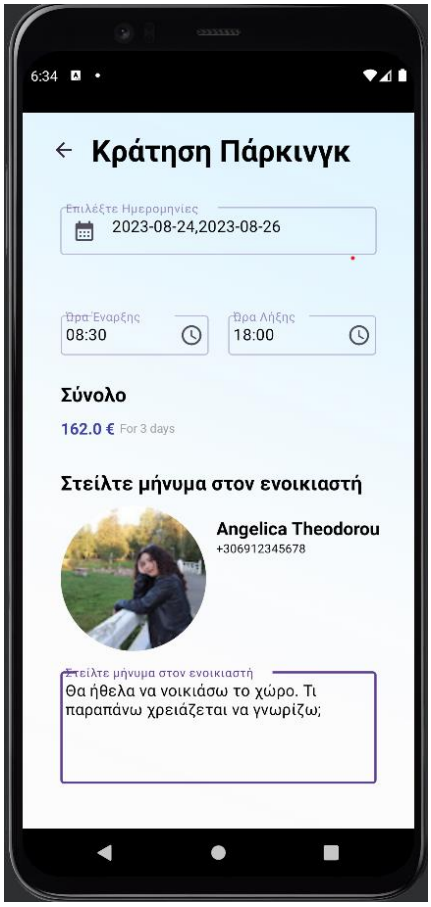


Εικόνα 51 ParkShare: Πληροφορίες θέσης πάρκινγκ, Ενοικιαστής

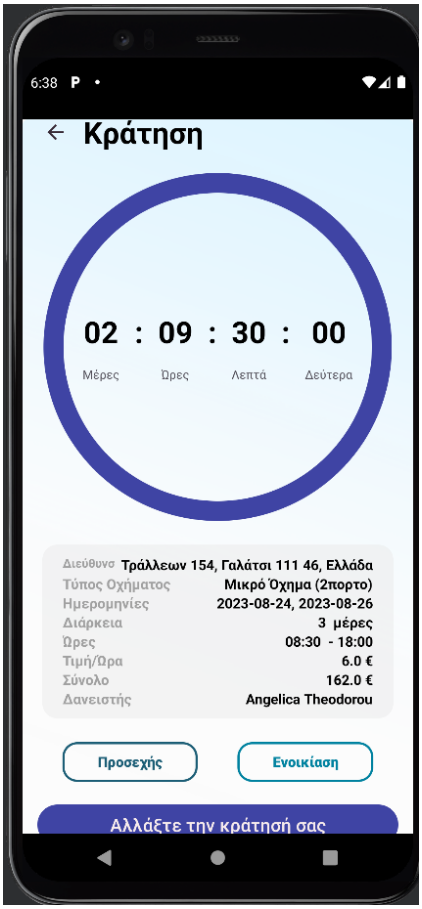
7.2.5 Κράτηση θέσης στάθμευσης

Όλοι οι χρήστες της εφαρμογής μπορούν να επιλέξουν ένα χώρο στάθμευσης και να κάνουν κράτηση αυτού. Στη φόρμα συμπληρώνουν τις μέρες και τις ώρες που θα ήθελαν να νοικιάσουν τη θέση και στέλνουν ένα μήνυμα στον ενοικιαστή έτσι ώστε να προβούν σε επικοινωνία και να διευθετήσουν οποιοδήποτε θέμα χρειάζεται. Ακόμη, μπορούν να ελέγξουν ή και να αλλάξουν τις κρατήσεις τους αφού τις κάνουν.

Εικόνα 52 ParkShare: Πληροφορίες θέσης πάρκινγκ, Οδηγός



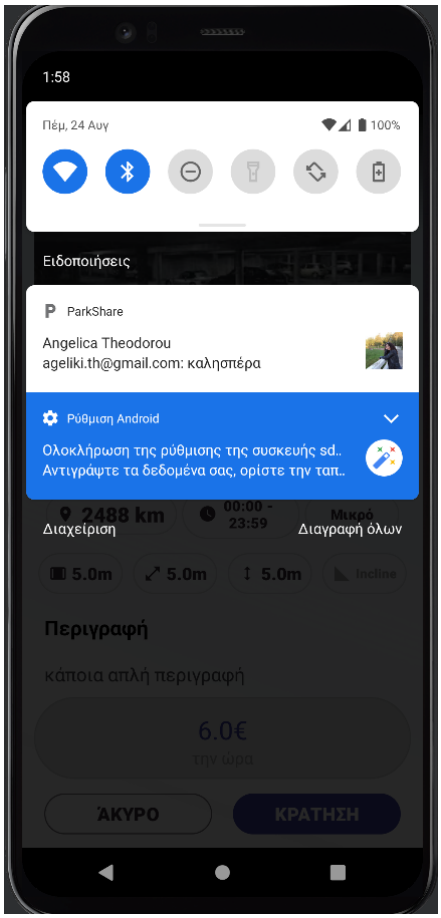
Εικόνα 53 ParkShare: Φόρμα κράτησης θέσης πάρκινγκ



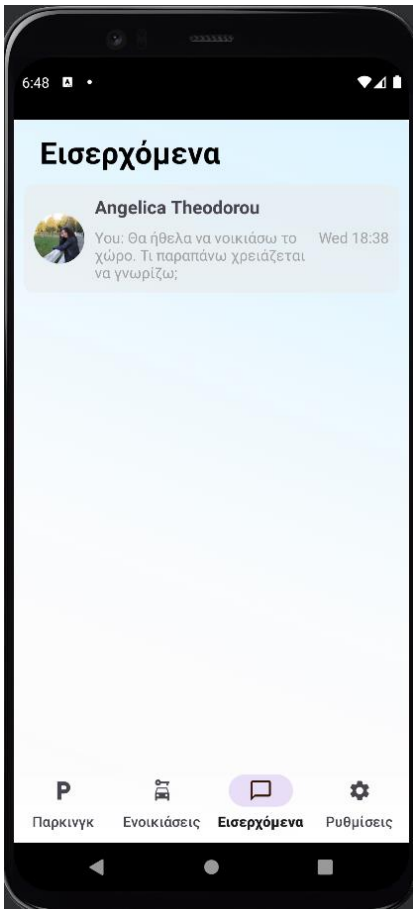
Εικόνα 54 ParkShare: Πληροφορίες κράτησης

7.2.6 Εισερχόμενα

Αφού γίνουν οι κρατήσεις οι χρήστες μπορούν να δουν τις συνομιλίες που έχουν κάνει με τους άλλους χρήστες από ή στους οποίους νοικιάζουν τις θέσεις πάρκινγκ στην επιλογή «Εισερχόμενα» της κάτω μπάρας. Ακόμη, για τις νέες κρατήσεις και τα νέα μηνύματα οι χρήστες ενημερώνονται με ειδοποιήσεις από την εφαρμογή.



Εικόνα 55 ParkShare: Ειδοποίηση μηνύματος από την εφαρμογή

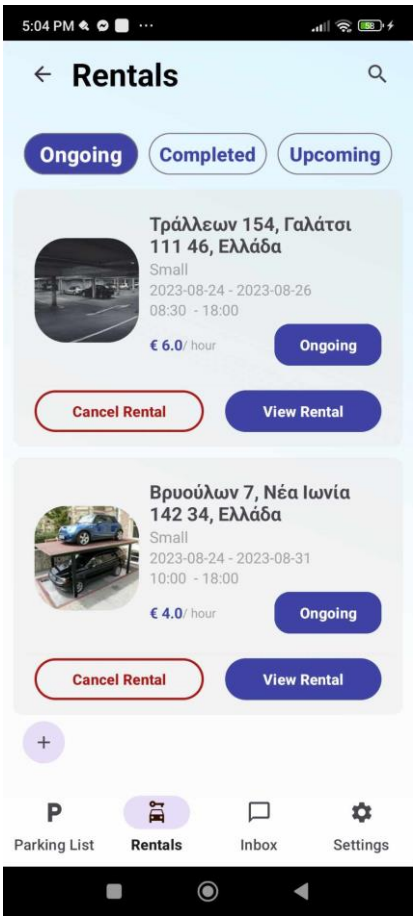


Εικόνα 56 ParkShare: Εισερχόμενα

Εικόνα 57 ParkShare: Συνομιλία

7.2.7 Ενοικιάσεις

Επιλέγοντας τις «Ενοικιάσεις» στην κάτω μπάρα οι χρήστες έχουν πρόσβαση σε όλες τις ενοικιάσεις που έχουν διεξάγει, μπορούν να παρακολουθούν το ιστορικό των κρατήσεών τους και να τις επεξεργάζονται.

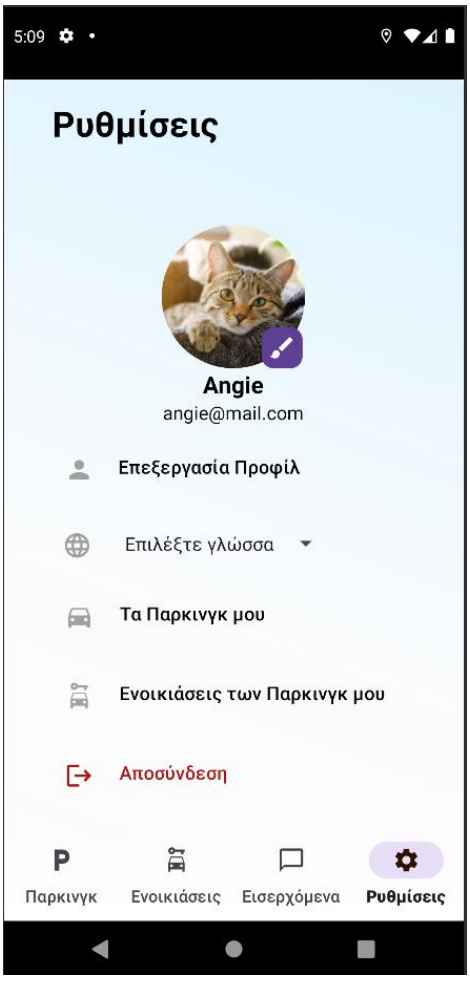


Εικόνα 58 ParkShare: Τρέχουσες Ενοικιάσεις, Οδηγοί

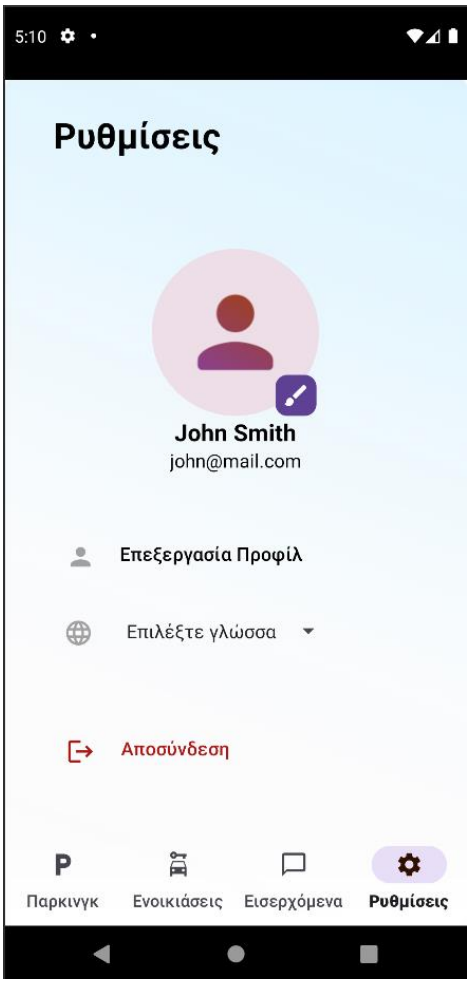
Εικόνα 59 ParkShare: Πληροφορίες κράτησης, Οδηγοί

7.2.8 Ρυθμίσεις

Επιλέγοντας τις «Ρυθμίσεις» στην κάτω μπάρα οι χρήστες έχουν τη δυνατότητα να κάνουν διάφορες αλλαγές στην εφαρμογή, ανάλογα με τις προτιμήσεις τους, καθώς και να επεξεργαστούν το προφίλ τους και να παρακολουθήσουν τις θέσεις παρκινγκ και τις κρατήσεις αυτών, εφόσον είναι ενοικιαστές.



Εικόνα 60 ParkShare: Ρυθμίσεις, Ενοικιαστές

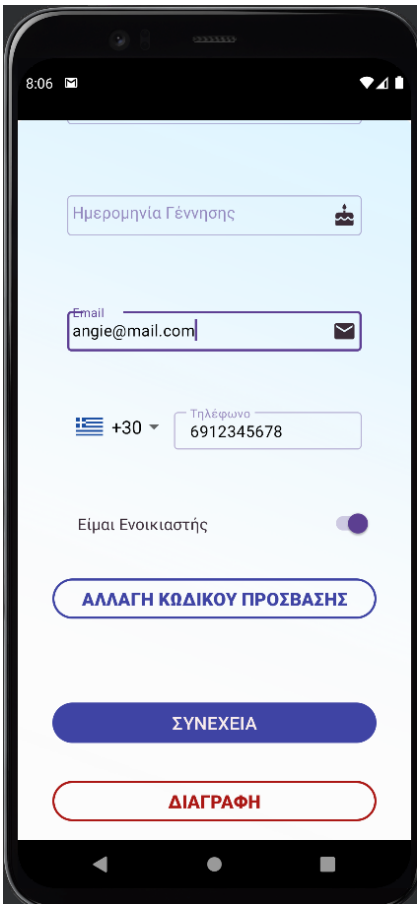


Εικόνα 61 ParkShare: Ρυθμίσεις, Οδηγοί

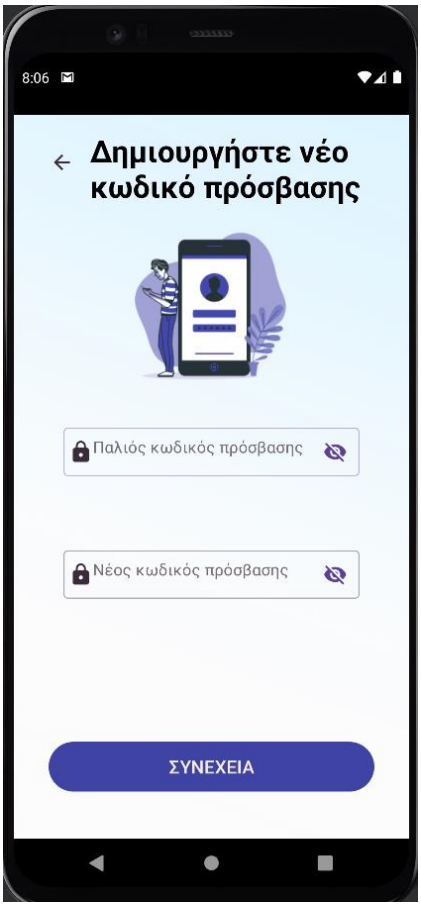
Προφίλ

Στην πρώτη επιλογή «Επεξεργασία Προφίλ», οι χρήστες μπορούν να επεξεργαστούν τις πληροφορίες του λογαριασμού τους, να αλλάξουν εικόνα προφίλ, αλλάξουν κωδικό πρόσβασης, κ.ο.κ.

Εικόνα 62 ParkShare: Οθόνη επεξεργασίας προφίλ 1



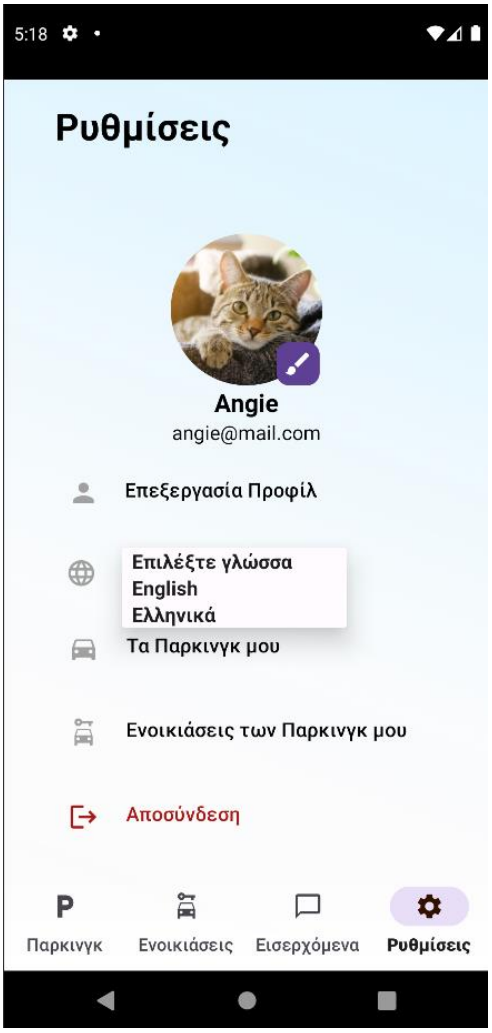
Εικόνα 63 ParkShare: Οθόνη επεξεργασίας προφίλ 2



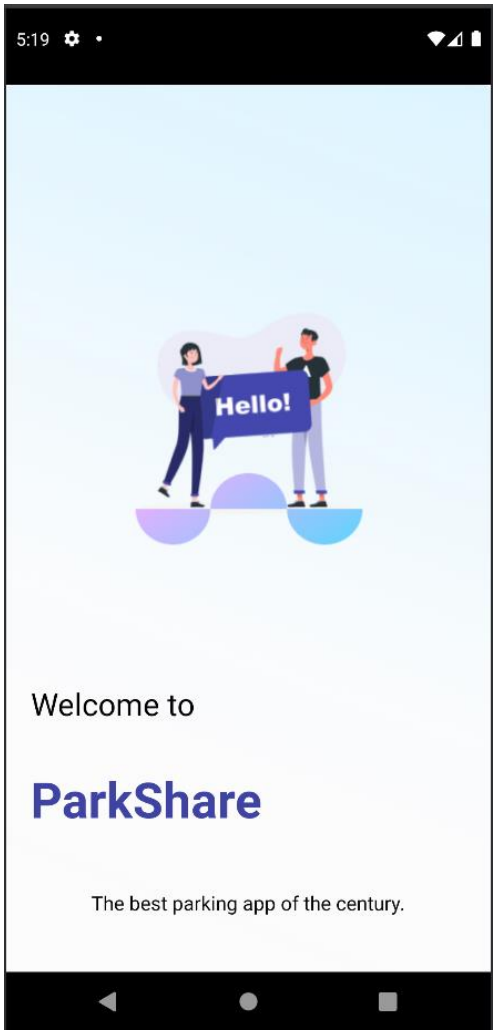
Εικόνα 64 ParkShare: Οθόνη αλλαγής κωδικού πρόσβασης

Γλώσσες εφαρμογής

Στη δεύτερη επιλογή «Επιλέξτε γλώσσα», οι χρήστες μπορούν να αλλάξουν τη γλώσσα της εφαρμογής από Αγγλικά σε Ελληνικά και αντίστροφα.



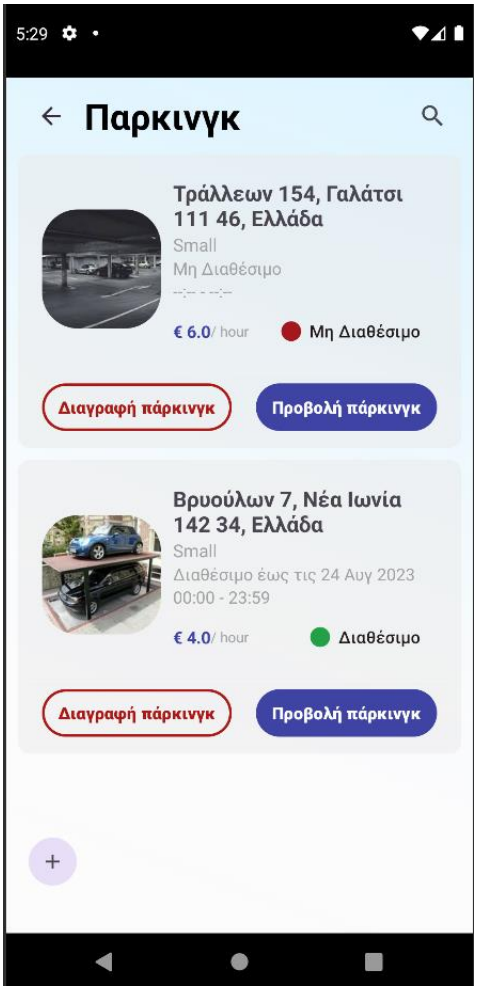
Εικόνα 65 ParkShare: Αλλαγή γλώσσας



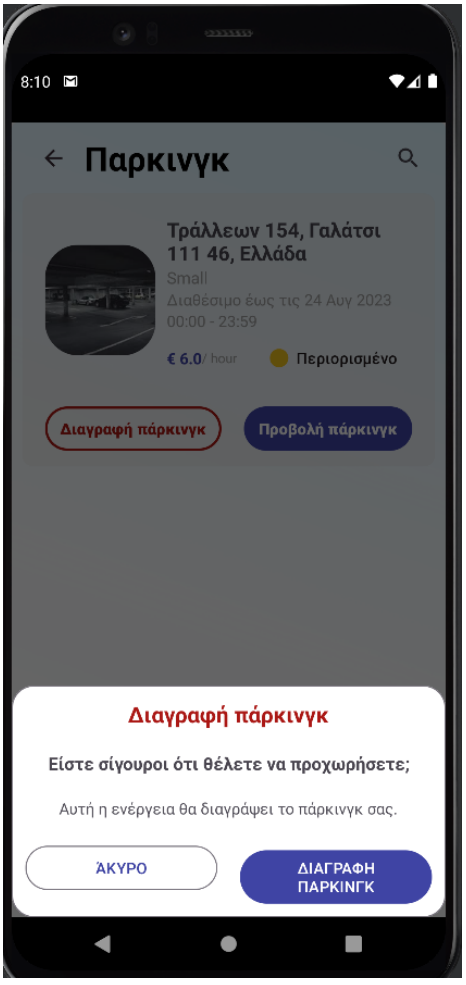
Εικόνα 66 ParkShare: Η εφαρμογή στα αγγλικά

Οι χώροι στάθμευσής μου

Εφόσον ο χρήστης είναι ενοικιαστής, η τρίτη επιλογή που εμφανίζεται στις ρυθμίσεις είναι η «Τα παρκινγκ μου». Εδώ εμφανίζονται οι χώροι στάθμευσης που έχει καταγράψει στην εφαρμογή ο ενοικιαστής προς ενοικίαση στους οδηγούς. Δίνεται η δυνατότητα προβολής, επεξεργασίας και διαγραφής του παρκινγκ από το σύστημα.



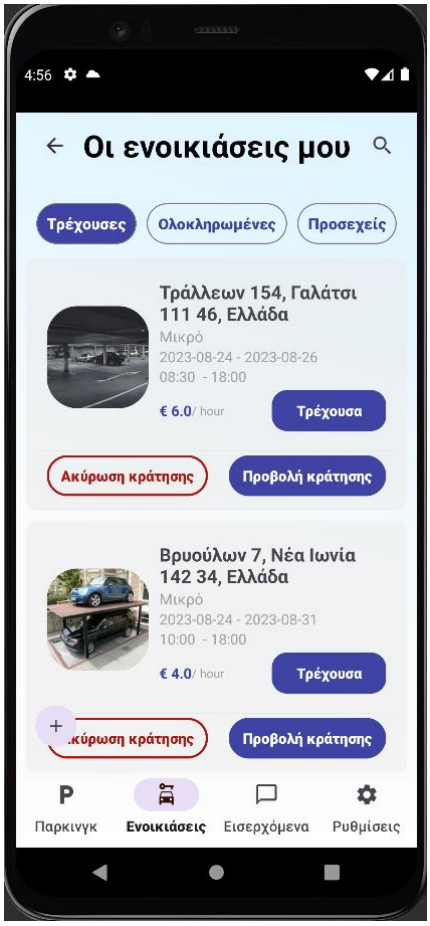
Εικόνα 67 ParkShare: Τα παρκινγκ μου



Εικόνα 68 ParkShare: Διαγραφή παρκινγκ

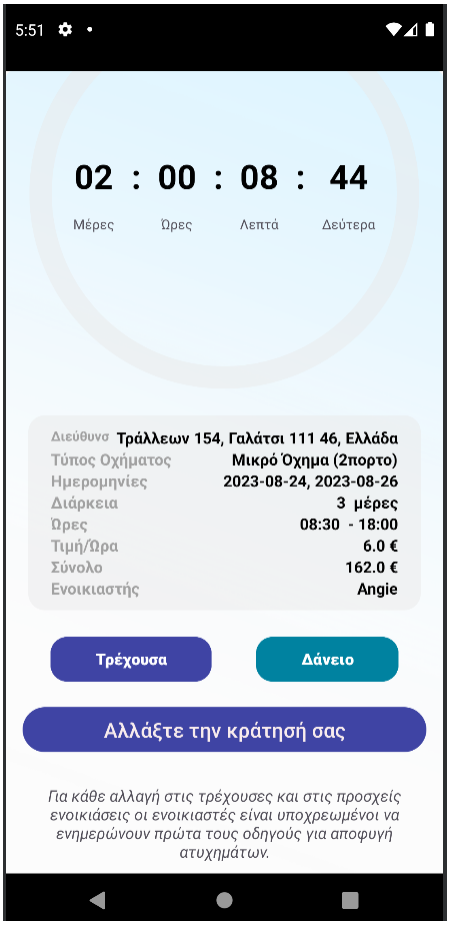
Οι ενοικιάσεις των χώρων μου

Αντίστοιχα, εφόσον ο χρήστης είναι ενοικιαστής στην τέταρτη επιλογή των ρυθμίσεων μπορεί να παρακολουθήσει τις ενοικιάσεις που έχουν γίνει από άλλους οδηγούς στους χώρους που έχει καταγράψει. Δίνεται η δυνατότητα προβολής, επεξεργασίας και ακύρωσης της κράτησης, μετά από τη συνομιλία με τον εκάστοτε οδηγό.



Εικόνα 69 ParkShare: Τρέχουσες Ενοικιάσεις, Ενοικιαστές

Εικόνα 70 ParkShare: Προσεχείς Ενοικιάσεις, Ενοικιαστές



Εικόνα 71 ParkShare: Πληροφορίες κράτησης, Ενοικιαστής

ΚΕΦΑΛΑΙΟ 7°: ΣΥΜΠΕΡΑΣΜΑΤΑ ΚΑΙ ΜΕΛΛΟΝΤΙΚΕΣ ΕΠΕΚΤΑΣΕΙΣ

8.1 ΜΕΛΛΟΝΤΙΚΕΣ ΕΠΕΚΤΑΣΕΙΣ

Η εργασία αυτή είχε διάφορες προκλήσεις, ωστόσο οι περισσότερες απαιτήσεις του πρότζεκτ ήλθαν εις πέρας με επιτυχία. Εντούτοις, υπάρχει πάντοτε χώρος για βελτιώσεις και επεκτάσεις των δυνατοτήτων της εφαρμογής αυτής.

Σκοπός της εργασίας αυτής ήταν η υλοποίηση μιας λύσης στο πρόβλημα αναζήτησης θέσεων στάθμευσης μέσω της ενοικίασης προσωπικών χώρων παρκινγκ. Χάρη στον τρόπο σχεδίασης της εφαρμογής τόσο σε Backend όσο και σε Mobile επίπεδο (πρότυπα σχεδίασης MVT και MVVM) υπάρχει η δυνατότητα επέκτασής της χωρίς να επηρεάζονται οι ως τώρα λειτουργίες της.

Η εφαρμογή θα μπορούσε να λάβει βελτιώσεις σε ό,τι αφορά την ασφάλεια των προσωπικών δεδομένων των χρηστών συγκεκριμένα για τις συνομιλίες μεταξύ των χρηστών και τις εικόνες που μοιράζονται στο σύστημα. Στην παρούσα φάση χρησιμοποιείται ένας απλός ασύγχρονος server για την ανταλλαγή των μηνυμάτων και η κύρια βάση δεδομένων για την αποθήκευσή τους. Σε επόμενα στάδια θα μπορούσε να βελτιωθεί η ασφάλεια των συνομιλιών εφαρμόζοντας end-to-end encryption (*E2EE*). Αντίστοιχα, για την αποθήκευση των εικόνων στο cloud χρησιμοποιείται η Firebase Storage, ωστόσο θα μπορούσε να χρησιμοποιείται κάποιος άλλος πάροχος αποθήκευσης αρχείων όπως οι υπηρεσίες της Microsoft Azure (Azure Files), της AWS (Amazon EFS), και άλλα. Ακόμη, θα ήταν ευνοϊκό η ανάρτηση των API υπηρεσιών να γίνει σε δημόσιο server, ο οποίος θα έχει τη δυνατότητα να παραμένει ενεργός, χωρίς να απαιτεί διαρκείς κινήσεις από το διαχειριστή του. Αφού, πραγματοποιηθεί αυτό, θα μπορούσαν να γίνουν βελτιώσεις στην ταχύτητα ανταπόκρισης από το server για ταχύτερη εξυπηρέτηση των χρηστών της εφαρμογής και βέλτιστη επικοινωνία μεταξύ Mobile και Back End. Επιπρόσθετα, λαμβάνοντας υπόψιν τα διάφορα λογισμικά κινητών τηλεφώνων που υπάρχουν σήμερα, κρίνεται απαραίτητη η επέκταση της λειτουργικότητας της εφαρμογής σε άλλα λογισμικά, πέραν του Android, όπως είναι το iOS. Στη συνέχεια, πρεσβεύοντας την ιδέα της ανάγκης εκσυγχρονισμού των παλαιών τεχνολογιών στα χρόνια των μεγάλων εξελίξεων και ως επέκταση στις ήδη υπάρχουσες λειτουργίες της εφαρμογής, θα μπορούσαν να προστεθούν περαιτέρω υπηρεσίες για τους χρήστες στην εφαρμογή. Ιδανικά, η εφαρμογή αυτή θα μπορούσε να γίνει ένα εργαλείο με μεγάλη χρηστικότητα από τους ενοικιαστές, δίνοντάς τους την επιλογή δυναμικού υπολογισμού κοστολόγησης, αυτόματης περιγραφής και προώθησης του χώρου τους, καθιστώντας έτσι πιο εύκολη την αναζήτηση θέσεων στάθμευσης και για τους οδηγούς.

8.2 ΣΥΜΠΕΡΑΣΜΑΤΑ

Κατά την υλοποίηση της εν λόγω μεταπτυχιακής διατριβής, υιοθετήθηκε μια μεθοδική προσέγγιση για τη δημιουργία και ανάπτυξη μιας εφαρμογής που σχεδιάστηκε για τον εντοπισμό των θέσεων στάθμευσης. Στο πλαίσιο της ανάπτυξης αξιοποιήθηκαν σύγχρονες τεχνολογίες, διευκολύνοντας την υλοποίηση ενός προηγμένου λογισμικού προσαρμοσμένου για την ενοικίαση θέσεων στάθμευσης.

Η δημιουργία της εφαρμογής βασίστηκε στις γνώσεις που αποκτήθηκαν από το μεταπτυχιακό πρόγραμμα, δίνοντας προτεραιότητα στις ανάγκες των χρηστών, την ατομική υποστήριξη και την ασφάλεια. Συνδυάζοντας τις θεωρητικές γνώσεις με την πρακτική εφαρμογή, δημιουργήθηκε μια πλατφόρμα που αντιμετωπίζει αποτελεσματικά το πρόβλημα εύρεσης χώρων παρκινγκ.

Κατά τη διάρκεια της φάσης ανάπτυξης, ενσωματώθηκαν νέες τεχνικές και δεξιότητες, διευκολύνοντας την εφαρμογή και την απρόσκοπτη ολοκλήρωσή τους στο έργο. Αυτή η προσπάθεια ενίσχυσε τη δυνατότητα αντιμετώπισης προκλήσεων και το σχεδιασμό λύσεων, με έμφαση στην καινοτομία και τη βελτίωση της εμπειρίας των χρηστών.

Αυτή η εφαρμογή ενσαρκώνει το αποτέλεσμα μιας μεθοδικής έρευνας, ανάπτυξης και εκπαίδευσης. Η επένδυση στην τεχνολογία και στη μάθηση αποδείχθηκε ένας επιτυχημένος δρόμος για τη δημιουργία μιας πλατφόρμας που ανταποκρίνεται στις απαιτήσεις της σύγχρονης πραγματικότητας, εστιάζοντας ειδικά στα προβλήματα της κυκλοφοριακής συμφόρησης και της αναζήτησης θέσεων στάθμευσης.

ΒΙΒΛΙΟΓΡΑΦΙΑ

1. Trista Lin, Hervé Rivano, Frédéric Le Mouël. A Survey of Smart Parking Solutions. IEEE Transactions on Intelligent Transportation Systems, IEEE, 2017, 18 (12), pp. 3229-3253. 10.1109/TITS.2017.2685143. hal-01501556
2. Car Parking Problem in Urban Areas, Causes and Solutions, Hossam El-Din I. S. Ahmed, Ph.D
3. The Parking Problem: A Game-Theoretic Solution, Giuseppe Calise, Aniello Murano, and Silvia Stranieri
4. JustPark app, https://www.justpark.com/
5. YourParkingSpace app, https://www.yourparkingspace.co.uk/
6. Stashbee, https://stashbee.com/
7. Similarweb: stashbee.com Traffic Analytics, Ranking Stats and Tech Stack, https://www.similarweb.com/website/stashbee.com/#overview
8. Similarweb: justpark.com Traffic Analytics, Ranking Stats and Tech Stack, https://www.similarweb.com/website/justpark.com/#overview
9. Similarweb: yourparkingspace.co.uk Traffic Analytics, Ranking Stats and Tech Stack, https://www.similarweb.com/website/yourparkingspace.co.uk/#overview
10. Park On My Drive app, https://www.parkonmydrive.com/
11. Parklet app, https://www.parklet.co.uk/
12. Spacer app, https://www.spacer.com/
13. ParkPnP app, https://parkpnp.com/ie/
14. Parking For Me app, https://www.parkingforme.com/
15. Minimum Viable Product (MVP), https://en.wikipedia.org/wiki/Minimum_viable_product
16. Τεχνολογία Ανάπτυξης Λογισμικού ως Υπηρεσίας: Armando Fox, David Patterson – Μέρος ΙΙ Ανάπτυξη Λογισμικού
17. PostgreSQL, https://www.postgresql.org/
18. Psycopg2-binary pip package documentation, https://pypi.org/project/psycopg2-binary/
19. Django Rest Framework Documentation, https://www.django-rest-framework.org/
20. Django MVT Architecture, https://www.javatpoint.com/django-mvt
21. Model-view-controller, https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller
22. Uvicorn implementation, https://www.uvicorn.org/
23. JSON Web Token, https://en.wikipedia.org/wiki/JSON_Web_Token

24. Firebase Cloud Messaging, https://firebase.google.com/docs/cloud-messaging
25. pyfcm pip package documentation, https://pypi.org/project/pyfcm/
26. Model-view-viewmodel, https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93viewmodel
27. Docker, https://www.docker.com/
28. Kotlin, https://kotlinlang.org/
29. Android MVVM Design Pattern, https://www.digitalocean.com/community/tutorials/android-mvvm-design-pattern
30. Retrofit, https://square.github.io/retrofit/
31. Firebase Storage, https://firebase.google.com/docs/storage
32. End-to-end encryption (E2EE), https://en.wikipedia.org/wiki/End-to-end_encryption
33. Azure Files, https://azure.microsoft.com/en-us/products/storage/files
34. Amazon Elastic File System (EFS), https://aws.amazon.com/efs/