



ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ
ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

«Εφαρμογή iOS Εύρεσης επί Πληρωμής Θέσης Στάθμευσης»



Φοιτητής

Αποστολίδης Θεοχάρης 185149

Επιβλέπων

Δρ. Κυριάκος Τσιακμάκης

Σεπτέμβριος 2026

Τίτλος Π. Ε. Εφαρμογή iOS Εύρεσης επί Πληρωμής Θέσης Στάθμευσης

Κωδικός: 25168

Φοιτητής: Αποστολίδης Θεοχάρης

Εισηγητής: Δρ Κυριάκος Τσιακμάκης

Ημερομηνία ανάληψης Π.Ε. 11-03-2025

Ημερομηνία περάτωσης Π.Ε. 25-08-2026

Βεβαιώνω ότι είμαι ο συγγραφέας αυτής της εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, έχω καταγράψει τις όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών, εικόνων και κειμένου, είτε αυτές αναφέρονται ακριβώς είτε παραφρασμένες. Επιπλέον, βεβαιώνω ότι αυτή η εργασία προετοιμάστηκε από εμένα προσωπικά, ειδικά ως πτυχιακή εργασία, στο Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του ΔΙ.ΠΑ.Ε.

Η παρούσα εργασία αποτελεί πνευματική ιδιοκτησία του φοιτητή Αποστολίδη Θεοχάρη που την εκπόνησε/αν. Στο πλαίσιο της πολιτικής ανοικτής πρόσβασης, ο συγγραφέας/δημιουργός εκχωρεί στο Διεθνές Πανεπιστήμιο της Ελλάδος άδεια χρήσης του δικαιώματος αναπαραγωγής, δανεισμού, παρουσίασης στο κοινό και ψηφιακής διάχυσης της εργασίας διεθνώς, σε ηλεκτρονική μορφή και σε οποιοδήποτε μέσο, για διδακτικούς και ερευνητικούς σκοπούς, άνευ ανταλλάγματος. Η ανοικτή πρόσβαση στο πλήρες κείμενο της εργασίας, δεν σημαίνει καθ' οιονδήποτε τρόπο παραχώρηση δικαιωμάτων διανοητικής ιδιοκτησίας του συγγραφέα/δημιουργού, ούτε επιτρέπει την αναπαραγωγή, αναδημοσίευση, αντιγραφή, πώληση, εμπορική χρήση, διανομή, έκδοση, μεταφόρτωση (downloading), ανάρτηση (uploading), μετάφραση, τροποποίηση με οποιονδήποτε τρόπο, τμηματικά ή περιληπτικά της εργασίας, χωρίς τη ρητή προηγούμενη έγγραφη συναίνεση του συγγραφέα/δημιουργού.

Η έγκριση της διπλωματικής εργασίας από το Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του Διεθνούς Πανεπιστημίου της Ελλάδος, δεν υποδηλώνει απαραίτητως και αποδοχή των απόψεων του συγγραφέα, εκ μέρους του Τμήματος.

Πρόλογος

Επέλεξα το θέμα αυτό γιατί είναι κάτι που έχει πραγματική χρήση. Είναι μια εφαρμογή που έψαξα να βρω, αλλά το μόνο που βρήκα είναι ένα κενό στην αγορά. Αυτό το κενό με ώθησε να φτιάξω την εφαρμογή αυτή. Γιατί όμως εφαρμογή; Είναι απλό, στις μέρες μας το κινητό μας έχει γίνει προέκταση του χεριού μας. Κάνουμε τα πάντα από εκεί. Το μέλλον των προγραμμάτων και εφαρμογών είναι στις φορητές συσκευές.

Περίληψη

Το θέμα αυτής της πτυχιακής εργασίας είναι η ανάπτυξη μιας εφαρμογής iOS καθώς και ενός backend server για την εύρεση ιδιωτικών χώρων στάθμευσης στην Θεσσαλονίκη. Έχει την δυνατότητα δημιουργίας λογαριασμού, που χρησιμοποιείται για την αποθήκευση αγαπημένων και την βαθμολόγηση των χώρων στάθμευσης. Μπορεί επίσης να ενημερωθεί για το ωράριο, τις παροχές καθώς και σε πραγματικό χρόνο, τις ελεύθερες θέσεις στάθμευσης. Ταυτόχρονα δεν αποθηκεύει δεδομένα του χρήστη πέραν από τα απολύτως απαραίτητα για την λειτουργία της εφαρμογής, καθώς και το όνομα του χρήστη.

«Parking spot finder iOS application»

«Theocharis Apostolidis»

Abstract

An iOS application and a backend server that contains the private parkings of Thessaloniki, ratings, favorites and amenities per parking, with live update of the available parking spots, handicap access etc. Backend is not tracking any user actions except favorites or ratings because they are necessary for the correct use of the application.

Ευχαριστίες

Θα ήθελα να ευχαριστήσω τους γονείς μου, την κοπέλα μου, και τους φίλους μου που στάθηκαν στο πλευρό μου καθ' όλη τη διάρκεια της εκπόνησης της Πτυχιακής Εργασίας.

Περιεχόμενα

Πρόλογος.....	4
Περίληψη.....	5
Abstract	6
Ευχαριστίες	7
Περιεχόμενα	8
Κατάλογος Σχημάτων	12
Κατάλογος Πινάκων.....	13
Συντομογραφίες.....	14
Κεφάλαιο 1ο: Εισαγωγή	15
1.1 Το πρόβλημα	15
1.2 Το κενό στην αγορά	15
1.3 Έμπνευση	15
Κεφάλαιο 2ο: Αρχιτεκτονική και Τεχνολογίες Backend.....	16
2.1 Εισαγωγή στο Backend Σύστημα.....	16
2.1.1 Ο Κύκλος Ζωής του Αιτήματος (Request-Response Lifecycle).....	16
2.2 Γλώσσα Προγραμματισμού Backend: PHP	17
2.2.1 Εξέλιξη της PHP και Χαρακτηριστικά της Έκδοσης 8.x.....	17
2.2.2 Διαδικαστικός vs. Αντικειμενοστραφής Προγραμματισμός στην PHP.....	17
2.2.3 Ασφάλεια και Προστασία Δεδομένων στην PHP	18
2.2.4 Διαχείριση Μνήμης και OPcache	18
2.3 Σχεσιακό Σύστημα Διαχείρισης Βάσεων Δεδομένων: MySQL	18
2.3.1 Σύγκριση Storage Engines: InnoDB vs. MyISAM	19
2.3.2 Δημιουργία Ευρετηρίων (Database Indexing)	19
2.4 Περιβάλλον Τοπικού Διακομιστή: XAMPP & Apache	20
2.4.1 Αρχιτεκτονική του Apache HTTP Server	20
2.4.2 Η Εργαλειοθήκη του XAMPP και το phpMyAdmin.....	20
2.5 Αρχιτεκτονικό Πρότυπο REST (Representational State Transfer)	20
2.5.1 Οι 6 Βασικοί Περιορισμοί του REST	20
2.5.2 Σύγκριση Αρχιτεκτονικών: REST vs. SOAP vs. GraphQL.....	21
2.6 Μορφή Ανταλλαγής Δεδομένων: JSON	21
2.6.1 Σύγκριση JSON με XML και Protocol Buffers (Protobuf).....	22
Κεφάλαιο 3ο: Σχεδιασμός και Ανάλυση Βάσης Δεδομένων	23

3.1	Αρχές Σχεδιασμού και Κανονικοποίηση (Normalization)	23
3.2	Διάγραμμα Οντοτήτων-Συσχετίσεων (ER Diagram)	23
3.3	Αναλυτικό Λεξικό Δεδομένων (Data Dictionary).....	25
3.3.1	Πίνακας: “users”	25
3.3.2	Πίνακας: “parkings”.....	26
3.3.3	Πίνακας: “parking_amenities”	27
3.3.4	Πίνακας: “parking_schedule”	28
3.3.5	Πίνακας: “parking_images”	29
3.3.6	Πίνακας: “ratings”.....	29
3.3.7	Πίνακας: “favourites”	30
3.3.8	Πίνακας: “otp”	30
3.4	Ακεραιότητα Δεδομένων και Περιορισμοί (Referential Integrity Constraints)	30
3.5	Πρόγραμμα Ανάπτυξης / Απασφαλμάτωσης Βάσης.....	31
3.5.1	phpMyAdmin.....	31
3.5.2	MySQLWorkbench.....	31
Κεφάλαιο 4ο: Υλοποίηση REST API και Ασφάλεια.....		32
4.1	Ασφάλεια και Ταυτοποίηση Χρηστών.....	32
4.1.1	HTTP Basic Authentication	32
4.1.2	Κρυπτογράφηση Κωδικών με bcrypt.....	33
4.2	Επαλήθευση Λογαριασμού (Email Verification & OTP)	34
4.3	Αναλυτική Τεκμηρίωση των Endpoints	37
4.3.1	Endpoint: “/status.php” (Κατάσταση Διακομιστή).....	37
4.3.2	Endpoint: “/user.php” (Διαχείριση Χρηστών)	37
4.3.3	Endpoint: “/verify.php” (Επαλήθευση OTP)	38
4.3.4	Endpoint: “/parkings.php” (Αναζήτηση Πάρκινγκ).....	39
4.3.5	Endpoint: “/favourites.php” (Αγαπημένα)	40
4.3.6	Endpoint: “/rating.php” (Αξιολογήσεις)	41
4.3.7	Endpoint: “/change_password.php” (Αλλαγή Κωδικού)	42
4.3.8	Endpoint: “/forgot_password.php” (Ανάκτηση Κωδικού).....	42
Κεφάλαιο 5ο: Τεχνολογίες Ανάπτυξης Εφαρμογής / Οικοσύστημα Apple.....		43
5.1	Εισαγωγή στο Οικοσύστημα της Apple	43
5.1.1	Η Γλώσσα Προγραμματισμού Swift.....	44
5.2	Το Ολοκληρωμένο Περιβάλλον Ανάπτυξης Xcode.....	45
5.2.1	Βασικά Χαρακτηριστικά του Xcode.....	45
5.3	Απαιτήσεις και Περιορισμοί Ανάπτυξης σε Περιβάλλον Apple.....	46

5.3.1	Γιατί είναι απαραίτητο το macOS;	46
5.4	Σύγκριση Τεχνολογιών: UIKit εναντίον SwiftUI	47
5.4.1	UIKit: Το Παραδοσιακό Framework	47
5.4.2	SwiftUI: Το Σύγχρονο Δηλωτικό Framework	48
5.4.3	Σύγκριση Αρχιτεκτονικής UI	48
5.5	Γιατί η SwiftUI Επιλέχθηκε για το Έργο iParkThess	48
5.5.1	Στρατηγική Επιλογή της Apple (Future-Proofing)	49
5.5.2	Ταχύτητα Ανάπτυξης (Rapid Prototyping)	49
5.5.3	Previews και Real-time Debugging	49
5.5.4	Εγγενείς Βελτιστοποιήσεις και Απόδοση	50
5.5.5	Διαχείριση Ροής Δεδομένων (Property Wrappers)	50
5.5.6	Διαχείριση Μνήμης και Lifecycle	51
Κεφάλαιο 6ο:	Ανάλυση, Αρχιτεκτονική και Οθόνες της Εφαρμογής	52
6.1	Αρχιτεκτονική της Εφαρμογής (Clean Architecture & MVVM)	52
6.1.1	Τα Επίπεδα του Clean Architecture	52
6.1.2	Dependency Injection (Έγχυση Εξαρτήσεων)	53
6.2	Αρχιτεκτονικό Διάγραμμα Εφαρμογής	53
6.3	Περιγραφή Οθονών και Λειτουργικότητας	54
6.3.1	Splash View & Onboarding (SplashView / LobbyView)	54
6.3.2	Αρχική Οθόνη (HomeView)	55
6.3.3	Οθόνη Χάρτη (MapView)	58
6.3.4	Αναζήτηση και Λίστα (SearchView)	60
6.3.5	Πληροφορίες Χώρου Στάθμευσης (ParkingDetailsView / ParkingDetailsHalfView)	62
6.3.6	Σύστημα Αξιολόγησης (RateView)	64
6.3.7	Διαχείριση Λογαριασμού & Ασφάλεια	64
6.3.8	AccountView (Ρυθμίσεις)	71
6.4	Σύστημα Εντοπισμού Τοποθεσίας (CoreLocation & LocationManager)	74
6.4.1	Υλοποίηση του LocationManager	75
6.5	Ασφάλεια Δεδομένων στη Συσκευή (iOS Keychain)	76
Κεφάλαιο 7ο:	Συμπεράσματα / Προτάσεις Βελτίωσης	77
7.1	Συμπεράσματα	77
7.2	Προτάσεις Βελτίωσης	77
BIBΛΙΟΓΡΑΦΙΑ		78

Κατάλογος Σχημάτων

Εικόνα 2.1 Διάγραμμα ακολουθίας της εμφάνισης πινεζών πάρκινγκ στην εφαρμογή.....	17
Εικόνα 3.1 Διάγραμμα ER βάσης.....	24
Εικόνα 6.1 Sequence Diagram της Clean Architecture.....	53
Εικόνα 6.2 Splash οθόνη (Dark Theme)	54
Εικόνα 6.3 Splash οθόνη (Light Theme).....	55
Εικόνα 6.4 Αρχική οθόνη αποσυνδεδεμένου χρήστη (Dark Theme).....	56
Εικόνα 6.5 Αρχική οθόνη αποσυνδεδεμένου χρήστη (Light Theme)	57
Εικόνα 6.6 Αρχική οθόνη συνδεδεμένου χρήστη (Dark Theme).....	57
Εικόνα 6.7 Αρχική οθόνη συνδεδεμένου χρήστη (Light Theme)	58
Εικόνα 6.8 Οθόνη Χάρτη (Dark Theme)	58
Εικόνα 6.9 Οθόνη Χάρτη (Light Theme).....	59
Εικόνα 6.10 Οθόνη Αναζήτησης (Light Theme)	61
Εικόνα 6.11 Οθόνη Αναζήτησης (Dark Theme)	61
Εικόνα 6.12 Οθόνη Πληροφοριών παρκινγκ (Light Theme)	63
Εικόνα 6.13 Οθόνη Πληροφοριών παρκινγκ (Dark Theme).....	63
Εικόνα 6.14 Οθόνη Αξιολόγησης (Light Theme)	64
Εικόνα 6.15 Οθόνη Αξιολόγησης (Dark Theme).....	64
Εικόνα 6.16 Οθόνη Σύνδεσης (Light Theme).....	65
Εικόνα 6.17 Οθόνη Σύνδεσης (Dark Theme).....	66
Εικόνα 6.18 Οθόνη Εγγραφής (Light Theme)	67
Εικόνα 6.19 Οθόνη Εγγραφής (Dark Theme)	67
Εικόνα 6.20 Οθόνη OTP (Light Theme).....	68
Εικόνα 6.21 Οθόνη OTP (Dark Theme).....	69
Εικόνα 6.22 Οθόνη Ανάκτησης Λογαριασμού (Light Theme)	70
Εικόνα 6.23 Οθόνη Ανάκτησης Λογαριασμού (Dark Theme).....	70
Εικόνα 6.24 Οθόνη Προφίλ (Light Theme)	71
Εικόνα 6.25 Οθόνη Προφίλ (Dark Theme).....	72
Εικόνα 6.26 Οθόνη Αλλαγής Γλώσσας (Light Theme)	72
Εικόνα 6.27 Οθόνη Αλλαγής Γλώσσας (Dark Theme)	73
Εικόνα 6.28 Οθόνη Αλλαγής Κωδικού (Light Theme).....	73
Εικόνα 6.29 Οθόνη Αλλαγής Κωδικού (Dark Theme)	74
Εικόνα 6.30 Sequence Diagram της ενημέρωσης τοποθεσίας	75

Κατάλογος Πινάκων

Πίνακας Αρχιτεκτονική και Τεχνολογίες Backend.1 Τεχνικές διαφορές InnoDB από MyISAM	19
Πίνακας Σχεδιασμός και Ανάλυση Βάσης Δεδομένων.2 Πίνακας "users" της βάσης δεδομένων	25
Πίνακας Σχεδιασμός και Ανάλυση Βάσης Δεδομένων.3 Πίνακας "parkings" της βάσης δεδομένων..	26
Πίνακας Σχεδιασμός και Ανάλυση Βάσης Δεδομένων.4 Πίνακας "parking_amenities" της βάσης δεδομένων	27
Πίνακας Σχεδιασμός και Ανάλυση Βάσης Δεδομένων.5 Πίνακας "parking_schedule" της βάσης δεδομένων	28
Πίνακας Σχεδιασμός και Ανάλυση Βάσης Δεδομένων.6 Πίνακας "parking_images" της βάσης δεδομένων	29
Πίνακας Σχεδιασμός και Ανάλυση Βάσης Δεδομένων.7 Πίνακας "ratings" της βάσης δεδομένων.....	29
Πίνακας Σχεδιασμός και Ανάλυση Βάσης Δεδομένων.8 Πίνακας "favourites" της βάσης δεδομένων	30
Πίνακας Σχεδιασμός και Ανάλυση Βάσης Δεδομένων.9 Πίνακας "otp" της βάσης δεδομένων	30

Συντομογραφίες

Δ.Ε.	Διπλωματική Εργασία
ΔΙΠΑΕ	Διεθνές Πανεπιστήμιο Ελλάδος
Π.Ε.	Πτυχιακή Εργασία

Κεφάλαιο 1ο: Εισαγωγή

1.1 Το πρόβλημα

Η ευκολία που προσφέρει το αυτοκίνητο είναι μεγάλη, για αυτό και το προτιμάμε αντί για τα μέσα μαζικής μεταφοράς. Όμως στο κέντρο της Θεσσαλονίκης δεν υπάρχουν αρκετές ελεύθερες θέσεις για πάρκινγκ. Αυτό έχει σαν αποτέλεσμα την δημιουργία του προβλήματος της εύρεσης θέσης. Οι κάτοικοι στο κέντρο της Θεσσαλονίκης δεν έβρισκαν να παρκάρουν τα αυτοκίνητά τους, χρειαζόταν μια αλλαγή. Έτσι ο δήμος Θεσσαλονίκης πλέον έχει ορίσει στο κέντρο της πόλης τις θέσεις πάρκινγκ ως θέσεις για κατοίκους, και θέσεις για επισκέπτες. Μόνο που με αυτήν την αλλαγή αρκετοί επισκέπτες δεν βρίσκουν θέσεις, άρα πρέπει να στραφούν σε ιδιωτικά πάρκινγκ.

1.2 Το κενό στην αγορά

Κάπως πρέπει να βρουν ένα πάρκινγκ που να έχει άδειες θέσεις, να είναι ανοιχτό, πιθανώς να έχει και φορτιστή (αν είναι ηλεκτρικό το αυτοκίνητο), ή ακόμα και να είναι προσβάσιμο για άτομα με αναπηρία. Δημιουργείται λοιπόν μια ανάγκη για μια πλατφόρμα που συλλέγει αυτές τις πληροφορίες και τις παρέχει στον χρήστη.

1.3 Έμπνευση

Η έμπνευση για την υλοποίηση για μια τέτοια εφαρμογή βρίσκεται ήδη στην καθημερινότητά μας. Στα φαρμακεία υπάρχουν από έξω οθόνες που δείχνουν ένα χάρτη με τα κοντινότερα εφημερεύοντα φαρμακεία. Ένα άλλο παράδειγμα είναι η εφαρμογή “FuelGR” η οποία χρησιμοποιεί τα δεδομένα (τιμές βενζίνης/λίτρο) που ανεβάζουν τα πρατήρια βενζίνης στο “fuelprices.gr” [1] (σελίδα του υπουργείου ανάπτυξης και ανταγωνιστικότητας) και τα μετατρέπει σε χρήσιμη πληροφορία για τον χρήστη.

Κεφάλαιο 2ο: Αρχιτεκτονική και Τεχνολογίες Backend

2.1 Εισαγωγή στο Backend Σύστημα

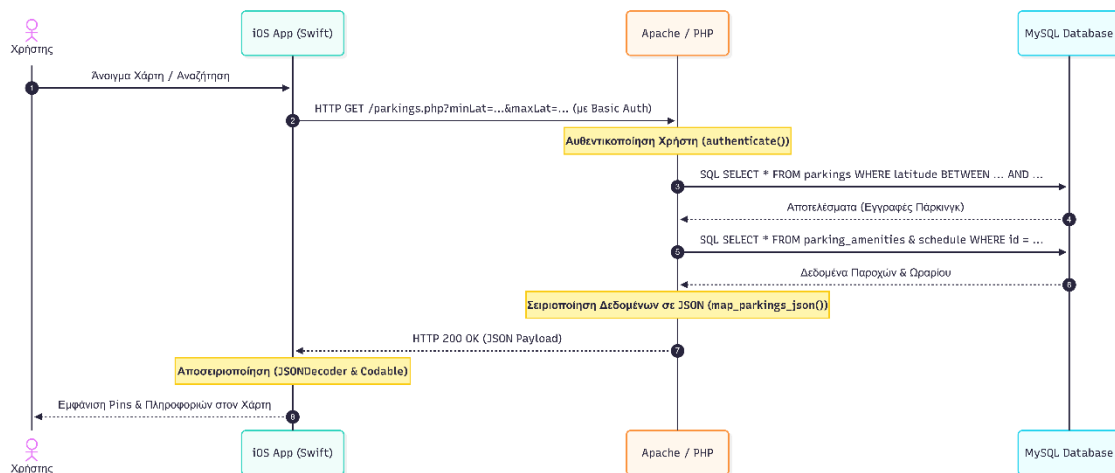
Η δομή της εφαρμογής iParkThess στηρίζεται στο κλασικό μοντέλο Πελάτη-Εξυπηρετητή (Client-Server Architecture). Στο μοντέλο αυτό, η iOS εφαρμογή λειτουργεί ως ο Πελάτης (Client), ο οποίος αναλαμβάνει την παρουσίαση της διεπαφής χρήστη (UI), τη λήψη των γεωγραφικών συντεταγμένων της συσκευής, την αλληλεπίδραση με τον χάρτη και την αποστολή αιτημάτων (requests) προς το Backend Server. Ο Backend Server αποτελεί την "καρδιά" του συστήματος, αναλαμβάνοντας την επεξεργασία των αιτημάτων, την αυθεντικοποίηση των χρηστών, την επικοινωνία με τη σχεσιακή βάση δεδομένων και την παροχή δεδομένων σε μορφή JSON.

2.1.1 Ο Κύκλος Ζωής του Αιτήματος (Request-Response Lifecycle)

Η επικοινωνία μεταξύ του client και του server ακολουθεί έναν αυστηρά καθορισμένο κύκλο ζωής. Για παράδειγμα ο τρόπος εμφάνισης των pins των πάρκινγκ στην εφαρμογή γίνεται ως εξής:

- **Αίτημα:** Ο χρήστης ανοίγει τον χάρτη στην iOS εφαρμογή. Η εφαρμογή ανιχνεύει τα γεωγραφικά όρια της οθόνης και στέλνει ένα HTTP GET αίτημα στο endpoint `"/parkings.php"` με τις αντίστοιχες συντεταγμένες ως παραμέτρους.
- **Δρομολόγηση & Επεξεργασία:** Ο Apache Web Server λαμβάνει το αίτημα, αναγνωρίζει ότι πρόκειται για αρχείο PHP και μεταβιβάζει την εκτέλεση στον διερμηνέα της PHP. Ο κώδικας PHP συνδέεται με τη βάση δεδομένων MySQL.
- **Ερώτηση Βάσης:** Εκτελείται ένα SQL query για την επιλογή των πάρκινγκ που βρίσκονται εντός των συντεταγμένων.
- **Δόμηση Απάντησης (Response):** Η PHP μετατρέπει τις εγγραφές της βάσης δεδομένων σε δομημένη μορφή JSON και τις επιστρέφει με HTTP Status Code 200 OK.
- **Απεικόνιση (Rendering):** Η iOS εφαρμογή κάνει parse το JSON και εμφανίζει τα pins των πάρκινγκ στον χάρτη.

Παρακάτω παρατίθεται το διάγραμμα ακολουθίας (Sequence Diagram) που αναπαριστά γραφικά αυτή τη ροή δεδομένων:



Εικόνα 2.1 Διάγραμμα ακολουθίας της εμφάνισης πινεζών πάρκινγκ στην εφαρμογή.

2.2 Γλώσσα Προγραμματισμού Backend: PHP

Για την ανάπτυξη της επιχειρηματικής λογικής του διακομιστή επιλέχθηκε η γλώσσα προγραμματισμού PHP (Hypertext Preprocessor) στην έκδοση 8.x. Η PHP είναι μια server-side γλώσσα σεναρίων (scripting language) ειδικά σχεδιασμένη για την ανάπτυξη δυναμικών ιστοσελίδων και διαδικτυακών εφαρμογών.

2.2.1 Εξέλιξη της PHP και Χαρακτηριστικά της Έκδοσης 8.x

Η PHP έχει εξελιχθεί σημαντικά από τις παλαιότερες εκδόσεις της, προσφέροντας πλέον χαρακτηριστικά που τη συγκαταλέγουν στις σύγχρονες γλώσσες προγραμματισμού backend. Στην έκδοση 8.x, εισήχθησαν κρίσιμες βελτιώσεις:

- **JIT (Just-In-Time) Compiler:** Επιτρέπει τη μεταγλώττιση τμημάτων του κώδικα σε assembly κατά το runtime, βελτιώνοντας σημαντικά την ταχύτητα εκτέλεσης υπολογιστικά απαιτητικών διεργασιών [2].
- **Typed Properties & Strict Types:** Επιβάλλει αυστηρό έλεγχο τύπων δεδομένων (type safety) στις ιδιότητες των κλάσεων και στις παραμέτρους των συναρτήσεων, μειώνοντας δραστικά τα runtime σφάλματα.
- **Union Types & Constructor Promotion:** Απλοποιούν τη σύνταξη του κώδικα και επιτρέπουν τη δήλωση πολλαπλών τύπων για μία μεταβλητή (π.χ. int / float).

2.2.2 Διαδικαστικός vs. Αντικειμενοστραφής Προγραμματισμός στην PHP

Αν και η PHP υποστηρίζει πλήρως το αντικειμενοστραφές μοντέλο (Object-Oriented Programming - OOP), για τις ανάγκες του iParkThess επιλέχθηκε μια υβριδική προσέγγιση. Οι βασικές λειτουργίες

δρομολόγησης και η επεξεργασία των HTTP requests γίνονται διαδικαστικά (procedural) για μέγιστη ταχύτητα και απλότητα, ενώ η σύνδεση με τη βάση δεδομένων και η διαχείριση των αποτελεσμάτων χρησιμοποιούν τις OOP δυνατότητες της κλάσης “mysqli” [3] και των prepared statements (“mysqli_stmt”).

2.2.3 Ασφάλεια και Προστασία Δεδομένων στην PHP

Η ασφάλεια του backend αποτελεί κρίσιμο παράγοντα. Η PHP παρέχει ενσωματωμένους μηχανισμούς για την αντιμετώπιση των πιο κοινών απειλών ασφάλειας:

- **SQL Injection:** Συμβαίνει όταν κακόβουλος κώδικας SQL εισάγεται σε πεδία εισόδου και εκτελείται από τη βάση δεδομένων. Στο iParkThess, η απειλή αυτή εξαλείφεται πλήρως με τη χρήση Prepared Statements. Τα δεδομένα που εισάγει ο χρήστης (π.χ. username, email) δεν συγχωνεύονται απευθείας στο SQL query, αλλά αποστέλλονται ξεχωριστά μέσω της συνάρτησης “bind_param()” [4], εμποδίζοντας τη βάση δεδομένων να τα ερμηνεύσει ως εκτελέσιμο κώδικα.
- **Cross-Site Scripting (XSS):** Αφορά την εισαγωγή κακόβουλων scripts που εκτελούνται στον client. Στην εφαρμογή μας, γίνεται αυστηρό φιλτράρισμα και sanitization των εισερχόμενων δεδομένων (π.χ. κατά την εγγραφή ή την αξιολόγηση) μέσω της PHP για την αφαίρεση HTML tags και ειδικών χαρακτήρων.
- **Ασφαλές Hashing Κωδικών:** Οι κωδικοί πρόσβασης δεν αποθηκεύονται ποτέ ως απλό κείμενο. Χρησιμοποιείται η συνάρτηση “password_hash()” [4] με τον αλγόριθμο bcrypt, ο οποίος παράγει ένα μοναδικό hash 60 χαρακτήρων που περιέχει αυτόματο salt. Αυτό καθιστά αδύνατη την αποκρυπτογράφηση των κωδικών ακόμη και σε περίπτωση διαρροής της βάσης δεδομένων.

2.2.4 Διαχείριση Μνήμης και OPcache

Η PHP διαχειρίζεται αυτόματα τη μνήμη, αποδεσμεύοντας τους πόρους μετά το τέλος κάθε request. Για τη βελτιστοποίηση της απόδοσης, χρησιμοποιείται το OPcache (Optimizer Plus) [4]. Το OPcache αποθηκεύει τον προ-μεταγλωττισμένο byte-code των αρχείων PHP στη μνήμη RAM. Με αυτόν τον τρόπο, ο server δεν χρειάζεται να διαβάζει και να αναλύει τα αρχεία PHP από τον δίσκο σε κάθε HTTP αίτημα, μειώνοντας τον χρόνο απόκρισης.

2.3 Σχεσιακό Σύστημα Διαχείρισης Βάσεων Δεδομένων: MySQL

Ως Σύστημα Διαχείρισης Βάσεων Δεδομένων (DBMS) επιλέχθηκε η **MySQL**. Πρόκειται για ένα σχεσιακό σύστημα (RDBMS) ανοικτού κώδικα, το οποίο χρησιμοποιεί τη γλώσσα δομημένων ερωτημάτων SQL (Structured Query Language) για τη διαχείριση των δεδομένων.

2.3.1 Σύγκριση Storage Engines: InnoDB vs. MyISAM

Η MySQL υποστηρίζει διάφορες μηχανές αποθήκευσης (storage engines) για τους πίνακες. Οι δύο πιο διαδεδομένες είναι η MyISAM και η InnoDB. Για την εφαρμογή iParkThess επιλέχθηκε αποκλειστικά η InnoDB λόγω των εξής τεχνικών διαφορών [5]:

Χαρακτηριστικό	InnoDB (Επιλογή μου)	MyISAM
Υποστήριξη Συναλλαγών (Transactions)	Ναι (ACID Compliant)	Όχι
Κλείδωμα (Locking)	Σε επίπεδο εγγραφής (Row-level)	Σε επίπεδο πίνακα (Table-level)
Ξένα Κλειδιά (Foreign Keys)	Ναι (Referential Integrity)	Όχι
Ανάκτηση από Σφάλμα (Crash Recovery)	Αυτόματη (μέσω Redo Logs)	Χειροκίνητη (επίφοβη για απώλεια δεδομένων)

Πίνακας Αρχιτεκτονική και Τεχνολογίες Backend.1 Τεχνικές διαφορές InnoDB από MyISAM

Η υποστήριξη ξένων κλειδιών είναι απαραίτητη για τη σχεσιακή δομή της βάσης μας, καθώς διασφαλίζει ότι δεν θα υπάρξουν ορφανές εγγραφές (π.χ. αξιολόγηση για χρήστη που δεν υπάρχει). Επίσης, το row-level locking της InnoDB επιτρέπει σε πολλούς χρήστες να ενημερώνουν ταυτόχρονα τις διαθέσιμες θέσεις ή να προσθέτουν αξιολογήσεις χωρίς να κλειδώνει ολόκληρος ο πίνακας, εξασφαλίζοντας υψηλή ταχύτητα σε συνθήκες ταυτόχρονης χρήσης (high concurrency).

2.3.2 Δημιουργία Ευρετηρίων (Database Indexing)

Για την επιτάχυνση των ερωτημάτων αναζήτησης, η MySQL χρησιμοποιεί Ευρετήρια (Indexes), τα οποία βασίζονται στη δομή δεδομένων B-Tree (Balanced Tree).

- Χωρίς ευρετήριο, η MySQL εκτελεί πλήρη σάρωση του πίνακα (Full Table Scan), η οποία έχει υπολογιστική πολυπλοκότητα $O(N)$, όπου N είναι ο αριθμός των εγγραφών.
- Με τη χρήση ευρετηρίων στα πεδία που χρησιμοποιούνται συχνά σε φίλτρα (όπως τα “id”, “latitude”, “longitude”), η πολυπλοκότητα μειώνεται σε $O(\log N)$, επιτρέποντας την ανάκτηση δεδομένων σε χιλιοστά του δευτερολέπτου ακόμη και σε πίνακες με εκατομμύρια εγγραφές.

Στη βάση δεδομένων του iParkThess, ευρετήρια δημιουργούνται αυτόματα στα Primary Keys (“id”), στα Foreign Keys και στα πεδία με περιορισμό “UNIQUE” (“username”, “email”).

2.4 Περιβάλλον Τοπικού Διακομιστή: XAMPP & Apache

Για την ανάπτυξη, εκτέλεση και δοκιμή του backend τοπικά στον υπολογιστή ανάπτυξης χρησιμοποιήθηκε η πλατφόρμα **XAMPP** σε συνδυασμό με τον διακομιστή Apache **HTTP Server**.

2.4.1 Αρχιτεκτονική του Apache HTTP Server

Ο Apache HTTP Server είναι ένας από τους πιο διαδεδομένους και αξιόπιστους διακομιστές ιστού παγκοσμίως. Χρησιμοποιεί Multi-Processing Modules (MPMs) για τη διαχείριση των εισερχόμενων συνδέσεων:

- **MPM Prefork:** Κάθε σύνδεση διαχειρίζεται από μια ξεχωριστή διεργασία (process). Είναι εξαιρετικά σταθερό και ασφαλές για τη λειτουργία της PHP, καθώς η αποτυχία μιας διεργασίας δεν επηρεάζει τις υπόλοιπες.
- **Thread-safe Execution:** Ο Apache συνεργάζεται άψογα με τον διερμηνέα της PHP, δημιουργώντας ένα περιβάλλον όπου τα αιτήματα επεξεργάζονται παράλληλα και αποδίδονται άμεσα.

2.4.2 Η Εργαλειοθήκη του XAMPP και το phpMyAdmin

Το XAMPP αποτελεί μια ενιαία διανομή που περιλαμβάνει τον Apache, τη MariaDB/MySQL, την PHP και την Perl. Η χρήση του απλοποιεί δραστικά τη διαμόρφωση του περιβάλλοντος εργασίας:

- **phpMyAdmin:** Παρέχει ένα ισχυρό, web-based γραφικό περιβάλλον για τη διαχείριση της MySQL. Μέσω αυτού υλοποιήθηκε η σχεδίαση των πινάκων, η εισαγωγή των αρχικών δεδομένων των πάρκινγκ της Θεσσαλονίκης και η εκτέλεση δοκιμαστικών queries εγγύτητας.
- **Localhost Environment:** Επιτρέπει την προσομοίωση ενός πραγματικού cloud διακομιστή στον τοπικό υπολογιστή, διευκολύνοντας την επικοινωνία με την iOS εφαρμογή μέσω της τοπικής διεύθυνσης IP ή του localhost.

2.5 Αρχιτεκτονικό Πρότυπο REST (Representational State Transfer)

Το API της εφαρμογής αναπτύχθηκε ακολουθώντας τις αρχές της αρχιτεκτονικής REST. Το REST είναι ένα αρχιτεκτονικό στυλ για τη σχεδίαση δικτυωμένων εφαρμογών, το οποίο βασίζεται στο πρωτόκολλο HTTP.

2.5.1 Οι 6 Βασικοί Περιορισμοί του REST

Για να χαρακτηριστεί ένα API ως REST-ful, πρέπει να ικανοποιεί έξι βασικούς περιορισμούς (REST Constraints):

- **Πελάτης-Εξυπηρετητής (Client-Server):** Σαφής διαχωρισμός διεπαφής (UI) και δεδομένων.

- **Απουσία Κατάστασης (Statelessness):** Κάθε αίτημα περιέχει όλα τα απαραίτητα στοιχεία για την εκτέλεσή του. Ο server δεν αποθηκεύει δεδομένα session στη μνήμη του. Αυτό επιτρέπει την εύκολη οριζόντια κλιμάκωση (horizontal scaling) του server προσθέτοντας νέους κόμβους χωρίς ανάγκη συγχρονισμού των sessions.
- **Δυνατότητα Προσωρινής Αποθήκευσης (Cacheability):** Οι απαντήσεις πρέπει να ορίζουν αν μπορούν να αποθηκευτούν στην προσωρινή μνήμη (cache) του client για τη μείωση των δικτυακών αιτημάτων.
- **Ομοιόμορφη Διεπαφή (Uniform Interface):** Χρήση standard HTTP μεθόδων (GET για ανάγνωση, POST για δημιουργία, DELETE για διαγραφή) και URIs για την ταυτοποίηση των πόρων (resources).
- **Συστηματοποίηση σε Επίπεδα (Layered System):** Ο client δεν μπορεί να γνωρίζει αν είναι συνδεδεμένος απευθείας με τον τελικό server ή με κάποιον ενδιάμεσο (load balancer, CDN, proxy).
- **Κώδικας κατ' Απαίτηση (Code on Demand - Προαιρετικό):** Η δυνατότητα του server να στέλνει εκτελέσιμο κώδικα (π.χ. JavaScript) στον client (δεν χρησιμοποιείται στην εφαρμογή μας για λόγους ασφαλείας και απόδοσης).

2.5.2 Σύγκριση Αρχιτεκτονικών: REST vs. SOAP vs. GraphQL

Κατά τον σχεδιασμό της επικοινωνίας εξετάστηκαν εναλλακτικές αρχιτεκτονικές:

- **SOAP (Simple Object Access Protocol):** Βασίζεται σε XML και απαιτεί αυστηρά WSDL συμβόλαια. Είναι εξαιρετικά "βαρύ" για mobile συσκευές λόγω του μεγάλου overhead των XML αρχείων και της πολυπλοκότητας του parsing.
- **GraphQL:** Επιτρέπει στον client να ζητά ακριβώς τα πεδία που χρειάζεται. Αν και μειώνει το over-fetching δεδομένων, απαιτεί πολύπλοκη παραμετροποίηση στον server, δυσκολεύει το caching και αυξάνει την κατανάλωση πόρων της CPU.
- **REST (Επιλογή μας):** Προσφέρει την ιδανική ισορροπία απλότητας, ευρείας υποστήριξης και χαμηλού overhead, καθιστώντας το την πλέον ενδεδειγμένη λύση για mobile εφαρμογές εύρεσης θέσεων στάθμευσης.

2.6 Μορφή Ανταλλαγής Δεδομένων: JSON

Η ανταλλαγή πληροφοριών μεταξύ του iOS client και του PHP backend πραγματοποιείται αποκλειστικά μέσω της μορφής JSON (JavaScript Object Notation).

2.6.1 Σύγκριση JSON με XML και Protocol Buffers (Protobuf)

Η επιλογή του JSON βασίζεται σε μια συγκριτική ανάλυση των εναλλακτικών προτύπων:

- JSON vs. XML: Το XML χρησιμοποιεί tags έναρξης και τέλους (π.χ. “<name>Parking</name>”), γεγονός που αυξάνει το μέγεθος του αρχείου κατά τουλάχιστον 30-40% σε σχέση με το JSON (π.χ. “{“name”:“Parking”}”). Επιπλέον, το parsing του XML απαιτεί περισσότερη επεξεργαστική ισχύ από τη mobile συσκευή.
- JSON vs. Protocol Buffers (Protobuf): Το Protobuf της Google είναι μια δυαδική (binary) μορφή σειριοποίησης, εξαιρετικά γρήγορη και με ελάχιστο μέγεθος [6]. Ωστόσο, είναι μια μη αναγνώσιμη μορφή από τον άνθρωπο (binary format), γεγονός που δυσκολεύει δραστικά την εκσφαλμάτωση (debugging) κατά την ανάπτυξη, και απαιτεί τη χρήση ειδικών compilers για τη δημιουργία κλάσεων σε Swift και PHP.

Το JSON προσφέρει την ιδανική χρυσή τομή: είναι εξαιρετικά ελαφρύ για το δίκτυο κινητής τηλεφωνίας, άμεσα αναγνώσιμο από τον προγραμματιστή για debugging μέσω εργαλείων όπως το Postman, και υποστηρίζεται natively από τη Swift μέσω του πρωτοκόλλου “Codable”.

Κεφάλαιο 3ο: Σχεδιασμός και Ανάλυση Βάσης Δεδομένων

3.1 Αρχές Σχεδιασμού και Κανονικοποίηση (Normalization)

Για τη διασφάλιση της σωστής λειτουργίας και της μελλοντικής επεκτασιμότητας του iParkThess, ο σχεδιασμός της βάσης δεδομένων βασίστηκε στις αρχές της Κανονικοποίησης (Normalization). Η κανονικοποίηση είναι μια συστηματική διαδικασία σχεδιασμού σχεσιακών βάσεων δεδομένων, η οποία στοχεύει στην ελαχιστοποίηση του πλεονασμού των δεδομένων (data redundancy) και στην αποτροπή ανωμαλιών εισαγωγής, ενημέρωσης και διαγραφής (insert, update, delete anomalies).

Η βάση δεδομένων σχεδιάστηκε έτσι ώστε να ικανοποιεί τις τρεις πρώτες βασικές κανονικές μορφές [7]:

- **Πρώτη Κανονική Μορφή (1NF):** Απαιτεί όλα τα πεδία ενός πίνακα να περιέχουν μόνο ατομικές (atomic) τιμές. Δεν επιτρέπονται επαναλαμβανόμενες ομάδες ή λίστες τιμών. Για τον λόγο αυτό, οι φωτογραφίες των πάρκινγκ δεν αποθηκεύονται ως λίστα σε ένα πεδίο του πίνακα “parkings”, αλλά μεταφέρθηκαν σε ξεχωριστό πίνακα “parking_images” όπου κάθε εγγραφή αντιστοιχεί σε μία φωτογραφία.
- **Δεύτερη Κανονική Μορφή (2NF):** Απαιτεί ο πίνακας να είναι σε 1NF και κάθε μη κλειδί πεδίο να εξαρτάται πλήρως από το πρωτεύον κλειδί (Primary Key) και όχι από μέρος αυτού. Στους πίνακες με σύνθετα κλειδιά, όπως ο πίνακας “ratings” (με PK τα “user_id, parking_id”), το πεδίο “rating” εξαρτάται πλήρως και από τις δύο οντότητες (ποιος χρήστης βαθμολόγησε ποιο πάρκινγκ).
- **Τρίτη Κανονική Μορφή (3NF):** Απαιτεί ο πίνακας να είναι σε 2NF και να μην υπάρχουν μεταβατικές εξαρτήσεις (transitive dependencies) μεταξύ πεδίων που δεν είναι κλειδιά. Για παράδειγμα, οι παροχές στάθμευσης (amenities) και το ωράριο λειτουργίας (schedule) διαχωρίστηκαν σε ξεχωριστούς πίνακες (“parking_amenities” και “parking_schedule”), καθώς αφορούν συγκεκριμένες επιχειρήσεις και δεν σχετίζονται άμεσα με τις γενικές πληροφορίες του πίνακα “parkings”.

3.2 Διάγραμμα Οντοτήτων-Συσχετίσεων (ER Diagram)

Το Διάγραμμα Οντοτήτων-Συσχετίσεων (Entity-Relationship Diagram) αποτυπώνει τις βασικές οντότητες του συστήματος, τις ιδιότητές τους (attributes) και τις μεταξύ τους λογικές συνδέσεις.

3.3 Αναλυτικό Λεξικό Δεδομένων (Data Dictionary)

3.3.1 Πίνακας: “users”

Αποθηκεύει τα στοιχεία των εγγεγραμμένων χρηστών που χρησιμοποιούν την iOS εφαρμογή.

Όνομα Πεδίου	Τύπος Δεδομένων	Περιορισμοί	Περιγραφή
“id”	INT	PRIMARY KEY, AUTO_INCREMENT, NOT NULL	Μοναδικός κωδικός χρήστη.
“username”	VARCHAR(64)	UNIQUE, NOT NULL	Μοναδικό όνομα χρήστη για την είσοδο (Basic Auth).
“passwordHash”	VARCHAR(255)	NOT NULL	Κρυπτογραφημένο password (bcrypt hash).
“email”	VARCHAR(255)	UNIQUE, NOT NULL	Ηλεκτρονική διεύθυνση για αποστολή OTP.
“full_name”	VARCHAR(255)	NULLABLE	Ονοματεπώνυμο χρήστη.
“verified”	BOOLEAN	NOT NULL, DEFAULT 0	1 αν ο λογαριασμός έχει επιβεβαιωθεί με OTP, αλλιώς 0.

Πίνακας Σχεδιασμός και Ανάλυση Βάσης Δεδομένων.2 Πίνακας "users" της βάσης δεδομένων

3.3.2 Πίνακας: “parkings”

Περιέχει τις βασικές πληροφορίες και τα real-time δεδομένα διαθεσιμότητας θέσεων για κάθε πάρκινγκ.

Όνομα Πεδίου	Τύπος Δεδομένων	Περιορισμοί	Περιγραφή
“id”	INT	PRIMARY KEY, AUTO_INCREMENT, NOT NULL	Μοναδικός κωδικός πάρκινγκ.
“name”	VARCHAR(64)	NOT NULL	Εμπορικός τίτλος της επιχείρησης.
“latitude”	DOUBLE(8,6)	NOT NULL	Γεωγραφικό πλάτος (συντεταγμένη GPS).
“longitude”	DOUBLE(8,6)	NOT NULL	Γεωγραφικό μήκος (συντεταγμένη GPS).
“address”	VARCHAR(64)	NOT NULL	Διεύθυνση πάρκινγκ.
“telephone”	VARCHAR(10)	NULLABLE	Τηλέφωνο επικοινωνίας (έως 10 ψηφία).
“max_car_spots”	INT	NOT NULL	Συνολικός αριθμός θέσεων αυτοκινήτων.
“available_car_spots”	INT	NOT NULL	Τρέχουσες ελεύθερες θέσεις αυτοκινήτων.
“max_motorbike_spots”	INT	NOT NULL	Συνολικός αριθμός θέσεων μοτοσυκλετών.
“available_motorbike_spots”	INT	NOT NULL	Τρέχουσες ελεύθερες θέσεις μοτοσυκλετών.

Πίνακας Σχεδιασμός και Ανάλυση Βάσης Δεδομένων.3 Πίνακας "parkings" της βάσης δεδομένων

3.3.3 Πίνακας: “parking_amenities”

Συνδέεται με σχέση 1:1 με τον πίνακα “parkings” και αποθηκεύει τις ειδικές παροχές του κάθε παρκινγκ.

Όνομα Πεδίου	Τύπος Δεδομένων	Περιορισμοί	Περιγραφή
“id”	INT	PRIMARY KEY, FOREIGN KEY (references “parkings.id” ON DELETE CASCADE)	Κωδικός πάρκινγκ.
“handicap_spots_available”	INT	NOT NULL	Τρέχουσες ελεύθερες θέσεις ΑμεΑ.
“max_handicap_spots”	INT	NOT NULL	Μέγιστες θέσεις ΑμεΑ.
“electric_chargers_available”	INT	NOT NULL	Τρέχοντες ελεύθεροι φορτιστές ηλεκτρικών αυτοκινήτων.
“max_electric_chargers”	INT	NOT NULL	Μέγιστοι φορτιστές ηλεκτρικών αυτοκινήτων.
“car_wash”	BOOLEAN	NOT NULL	Ένδειξη αν παρέχεται πλύσιμο.
“valet”	BOOLEAN	NOT NULL	Ένδειξη αν παρέχεται υπηρεσία παρκαδόρου.
“gas_station”	BOOLEAN	NOT NULL	Ένδειξη αν υπάρχει πρατήριο καυσίμων.

Πίνακας Σχεδιασμός και Ανάλυση Βάσης Δεδομένων.4Πίνακας "parking_amenities" της βάσης δεδομένων

3.3.4 Πίνακας: “parking_schedule”

Συνδέεται με σχέση 1:1 με τον πίνακα “parkings” και ορίζει το ωράριο λειτουργίας του κάθε πάρκινγκ.

Όνομα Πεδίου	Τύπος Δεδομένων	Περιορισμοί	Περιγραφή
“id”	INT	PRIMARY KEY, FOREIGN KEY (references “parkings.id” ON DELETE CASCADE)	Κωδικός πάρκινγκ.
“monday_open” / “monday_close”	VARCHAR(5)	NULLABLE	Ωρα έναρξης/λήξης της Δευτέρας (π.χ. '07:00' - '23:00'). Αν είναι NULL, είναι κλειστό.
“tuesday_open” / “tuesday_close”	VARCHAR(5)	NULLABLE	Ωρα έναρξης/λήξης της Τρίτης.
“wednesday_open” / “wednesday_close”	VARCHAR(5)	NULLABLE	Ωρα έναρξης/λήξης της Τετάρτης.
“thursday_open” / “thursday_close”	VARCHAR(5)	NULLABLE	Ωρα έναρξης/λήξης της Πέμπτης.
“friday_open” / “friday_close”	VARCHAR(5)	NULLABLE	Ωρα έναρξης/λήξης της Παρασκευής.
“saturday_open” / “saturday_close”	VARCHAR(5)	NULLABLE	Ωρα έναρξης/λήξης του Σαββάτου.
“sunday_open” / “sunday_close”	VARCHAR(5)	NULLABLE	Ωρα έναρξης/λήξης της Κυριακής.

Πίνακας Σχεδιασμός και Ανάλυση Βάσης Δεδομένων.5 Πίνακας "parking_schedule" της βάσης δεδομένων

3.3.5 Πίνακας: “parking_images”

Αποθηκεύει τα URL των εικόνων που σχετίζονται με το κάθε πάρκινγκ (σχέση 1:N).

Όνομα Πεδίου	Τύπος Δεδομένων	Περιορισμοί	Περιγραφή
“id”	INT	PRIMARY KEY, AUTO_INCREMENT, NOT NULL	Μοναδικός κωδικός εικόνας.
“name”	VARCHAR(64)	NOT NULL	Η διαδρομή (URL/path) της εικόνας στον server (π.χ. “/assets/1-1.png”).
“parking_id”	INT	FOREIGN KEY (references “parkings.id” ON DELETE CASCADE)	Κωδικός πάρκινγκ.

Πίνακας Σχεδιασμός και Ανάλυση Βάσης Δεδομένων.6 Πίνακας "parking_images" της βάσης δεδομένων

3.3.6 Πίνακας: “ratings”

Αποθηκεύει τις αξιολογήσεις των χρηστών (σχέση N:M μεταξύ χρηστών και πάρκινγκ).

Όνομα Πεδίου	Τύπος Δεδομένων	Περιορισμοί	Περιγραφή
“user_id”	INT	PRIMARY KEY, FOREIGN KEY (references “users.id” ON DELETE CASCADE)	Κωδικός χρήστη.
“parking_id”	INT	PRIMARY KEY, FOREIGN KEY (references “parkings.id” ON DELETE CASCADE)	Κωδικός πάρκινγκ.
“rating”	INT	NOT NULL	Βαθμολογία (κλίμακα 1 έως 5).

Πίνακας Σχεδιασμός και Ανάλυση Βάσης Δεδομένων.7 Πίνακας "ratings" της βάσης δεδομένων

3.3.7 Πίνακας: “favourites”

Αποθηκεύει τα αγαπημένα πάρκινγκ κάθε χρήστη (σχέση N:M μεταξύ χρηστών και πάρκινγκ).

Όνομα Πεδίου	Τύπος Δεδομένων	Περιορισμοί	Περιγραφή
“user_id”	INT	PRIMARY KEY, FOREIGN KEY (references “users.id” ON DELETE CASCADE)	Κωδικός χρήστη.
“parking_id”	INT	PRIMARY KEY, FOREIGN KEY (references “parkings.id” ON DELETE CASCADE)	Κωδικός πάρκινγκ.

Πίνακας Σχεδιασμός και Ανάλυση Βάσης Δεδομένων.8 Πίνακας "favourites" της βάσης δεδομένων

3.3.8 Πίνακας: “otp”

Αποθηκεύει τους προσωρινούς κωδικούς επαλήθευσης των χρηστών.

Όνομα Πεδίου	Τύπος Δεδομένων	Περιορισμοί	Περιγραφή
“user_id”	INT	PRIMARY KEY, FOREIGN KEY (references “users.id” ON DELETE CASCADE)	Κωδικός χρήστη.
“code”	INT	PRIMARY KEY, NOT NULL	Ο 6ψήφιος κωδικός μίας χρήσης (OTP).
“created_at”	TIMESTAMP	DEFAULT CURRENT_TIMESTAM P	Χρόνος δημιουργίας του κωδικού (χρησιμοποιείται για λήξη 10 λεπτών).

Πίνακας Σχεδιασμός και Ανάλυση Βάσης Δεδομένων.9 Πίνακας "otp" της βάσης δεδομένων

3.4 Ακεραιότητα Δεδομένων και Περιορισμοί (Referential Integrity Constraints)

Η διατήρηση της ακεραιότητας των δεδομένων της βάσης εξασφαλίζεται μέσω:

- Foreign Keys: Όλοι οι συνδεδεμένοι πίνακες χρησιμοποιούν ξένα κλειδιά που δείχνουν στους βασικούς πίνακες “users” και “parkings”.

- Cascade Delete: Σε όλα τα Foreign Keys έχει οριστεί η ιδιότητα “ON DELETE CASCADE”. Αυτό σημαίνει ότι αν διαγραφεί ένας λογαριασμός χρήστη, διαγράφονται αυτόματα τα αγαπημένα του, οι αξιολογήσεις του και οι κωδικοί OTP του. Αντίστοιχα, αν διαγραφεί ένα πάρκινγκ, διαγράφονται αυτόματα οι φωτογραφίες του, οι παροχές του, το ωράριό του, οι αξιολογήσεις και τα αγαπημένα που αναφέρονται σε αυτό. Αυτό αποτρέπει σφάλματα αναφοράς (orphan rows) και κρατά τη βάση καθαρή.
- Unique Constraints: Τα πεδία “username” και “email” στον πίνακα “users” έχουν δηλωθεί ως μοναδικά (“UNIQUE”) για να αποτραπεί η διπλή εγγραφή χρηστών με τα ίδια στοιχεία.

3.5 Πρόγραμμα Ανάπτυξης / Απασφαλμάτωσης Βάσης

Για την ανάπτυξη της βάσης δεδομένων καθώς και την απασφαλμάτωση (debugging) των σφαλμάτων που διαπιστώθηκαν κατά την ανάπτυξη / testing της iOS εφαρμογής iParkThess που είχε σχέση με την βάση δεδομένων, υπήρξαν δυο προγράμματα που σκέφτηκα να χρησιμοποιήσω.

3.5.1 phpMyAdmin

Το phpMyAdmin αποτελεί ένα ιδιαίτερα διαδεδομένο, ανοικτού κώδικα web-based εργαλείο για τη διαχείριση βάσεων δεδομένων MySQL και MariaDB. Χάρη στο φιλικό και εύχρηστο περιβάλλον που προσφέρει μέσω περιηγητή (browser), επιτρέπει στους προγραμματιστές να εκτελούν ερωτήματα SQL, να διαμορφώνουν δομές δεδομένων και να διαχειρίζονται αντίγραφα ασφαλείας χωρίς τη χρήση γραμμής εντολών.

Παράλληλα, διαδραματίζει καθοριστικό ρόλο στη λειτουργία σύγχρονων διαδικτυακών εφαρμογών, αποτελώντας βασικό κομμάτι του τεχνικού σκελετού πίσω από τη διαχείριση του WordPress. Μέσω αυτού εξασφαλίζεται η άμεση πρόσβαση, η συντήρηση και η παραμετροποίηση της βάσης δεδομένων στην οποία αποθηκεύεται το σύνολο του περιεχομένου, των χρηστών και των δομικών ρυθμίσεων του ιστότοπου.

3.5.2 MySQLWorkbench

Το MySQLWorkbench είναι ένα επίσημο, visual εργαλείο διαχείρισης και σχεδιασμού βάσεων δεδομένων MySQL. Σε σύγκριση με το phpMyAdmin, θεωρείται συχνά πιο εύχρηστο για τον αναπτυκτή, καθώς προσφέρει ένα ολοκληρωμένο περιβάλλον εργασίας με οπτική μοντελοποίηση (ER diagrams) και προηγμένη διαχείριση διακομιστών.

Παράλληλα, διαθέτει έναν πολύ πιο προηγμένο SQL editor, ο οποίος προσφέρει δυνατότητες όπως αυτόματη συμπλήρωση κώδικα (autocomplete), εντοπισμό σφαλμάτων και ανάλυση απόδοσης ερωτημάτων. Το σημαντικότερο πλεονέκτημά του είναι ότι πρόκειται για native εφαρμογή υπολογιστή (desktop client). Αυτό το καθιστά αισθητά πιο γρήγορο και αποδοτικό, καθώς αξιοποιεί απευθείας τους πόρους του συστήματος χωρίς τους περιορισμούς, τα όρια μνήμης ή τα timeouts ενός web browser.

Κεφάλαιο 4ο: Υλοποίηση REST API και Ασφάλεια

4.1 Ασφάλεια και Ταυτοποίηση Χρηστών

Η ασφάλεια των δεδομένων και η ταυτοποίηση των χρηστών αποτελεί θεμελιώδη απαίτηση για την εφαρμογή iParkThess. Για την προστασία των endpoints που εκτελούν κρίσιμες λειτουργίες (όπως η υποβολή αξιολογήσεων, η προσθήκη αγαπημένων και η αλλαγή κωδικών), χρησιμοποιήθηκαν οι εξής τεχνολογίες ασφάλειας:

4.1.1 HTTP Basic Authentication

Για κάθε αίτημα που απαιτεί ταυτοποίηση, η iOS εφαρμογή στέλνει τα στοιχεία σύνδεσης του χρήστη στην κεφαλίδα (header) “Authorization” του HTTP αιτήματος. Η μορφή της κεφαλίδας είναι η εξής:

“Authorization: Basic <Base64(Username:Password)>“

Στον διακομιστή (PHP), η αυθεντικοποίηση ελέγχεται μέσω της συνάρτησης “authenticate()”, η οποία ανακτά τις τιμές:

- PHP_AUTH_USER
- PHP_AUTH_PW

Στον κώδικα μεταφράζεται ως εξής:

```
if (!isset($_SERVER['PHP_AUTH_USER'])) {  
    return false;  
}  
  
$username = $_SERVER['PHP_AUTH_USER'];  
  
if (!isset($_SERVER['PHP_AUTH_PW'])) {  
    return false;  
}  
  
$password = $_SERVER['PHP_AUTH_PW'];
```

Η χρήση του Basic Authentication επιλέχθηκε λόγω της απλότητάς της και της stateless φύσης του REST API, καθώς ο server δεν χρειάζεται να διατηρεί sessions στη μνήμη του.

4.1.2 Κρυπτογράφηση Κωδικών με bcrypt

Οι κωδικοί πρόσβασης των χρηστών δεν αποθηκεύονται ποτέ σε μορφή απλού κειμένου (plain text) στη βάση δεδομένων. Κατά την εγγραφή ενός νέου χρήστη (μέσω της συνάρτησης “create_user()” [4]), ο κωδικός κρυπτογραφείται χρησιμοποιώντας τον ισχυρό αλγόριθμο κατακερματισμού (hashing) bcrypt μέσω της μεθόδου “password_hash()” [4]:

```
function create_user() {
    $body = file_get_contents('php://input');
    $data = json_decode($body, true);
    $username = $_SERVER['PHP_AUTH_USER'] ?? null;
    $password = $_SERVER['PHP_AUTH_PW'] ?? null;
    $email = $data['email'] ?? null;
    $name = $data['name'] ?? null;
    if ($username == null || $email == null || $password == null) {
        unprocessable();
    }
    $passwordHash = password_hash($password, PASSWORD_DEFAULT);
    $connection = connect();
    try {
        $stmt = $connection->prepare("insert into users(username, passwordHash, email, full_name) VALUES (?, ?, ?, ?)");
        $stmt->bind_param("ssss", $username, $passwordHash, $email, $name);
        $stmt->execute();
        header("HTTP/1.1 200 OK");
        header("Content-Type: application/json");
        echo '{"status": 200}';
        exit;
    } catch (mysqli_sql_exception $e) {
        if ($e->getCode() === 1062) {
            $errorMessage = $e->getMessage();
            if (str_contains($errorMessage, 'username') !== false) {
                conflict_user_exists();
            } elseif (str_contains($errorMessage, 'email') !== false) {
                conflict_email_exists();
            }
        } else {
            internal_error();
        }
    }
}
```

Κατά τη σύνδεση του χρήστη, η επαλήθευση γίνεται με την μέθοδο “password_verify()” όπου συγκρίνεται το hash που έχει αποθηκευτεί στην βάση με το hash που θα προκύψει από τον κωδικό που προσφέρει η κλήση και μόνο άμα είναι ίδια αποδέχεται την κλήση.

```
$queryString = "select passwordHash from users where username = '" . $username . "'";  
$user = $connection->query($queryString);  
if ($user->num_rows == 1) {  
    $db_user = $user->fetch_assoc();  
    $db_hash = $db_user['passwordHash'];  
    return password_verify($password, $db_hash);  
} else {  
    return false;  
}
```

Ο αλγόριθμος bcrypt ενσωματώνει αυτόματα έναν τυχαίο κωδικό “άλατος” (salt) για κάθε εγγραφή και είναι ανθεκτικός σε επιθέσεις brute-force και rainbow tables λόγω του ρυθμιζόμενου κόστους υπολογισμού του. Η μέθοδος “password_verify()” [4] είναι ικανή να διαχωρίσει τον κωδικό “άλατος” και να συγκρίνει την πραγματική πληροφορία πίσω από τα hash και τους κωδικούς που προσφέρονται.

4.2 Επαλήθευση Λογαριασμού (Email Verification & OTP)

Για την αποτροπή της δημιουργίας ψεύτικων λογαριασμών και την επιβεβαίωση της ταυτότητας των χρηστών, υλοποιήθηκε ένας μηχανισμός One-Time Password (OTP).

1. Δημιουργία OTP: Κατά την εγγραφή, παράγεται ένας τυχαίος εξαψήφιος κωδικός μέσω της μεθόδου “rand(100000, 999999)”.

2. Αποστολή Email: Ο κωδικός αποστέλλεται στο email του χρήστη μέσω της μεθόδου “mail()” [4] (στο αρχείο “sendmail.php”) με κατάλληλες κεφαλίδες UTF-8 για την αποφυγή φιλτραρίσματος ως ανεπιθύμητη αλληλογραφία (spam).

```
function sendmail($recipient) {  
    // Generating a random 6-digit verification code  
    $verification_code = rand(100000, 999999);  
  
    $recipient_mail = get_user_email($recipient);  
    $subject = "Your Verification Code";  
    $message = "Your secure verification code is: " . $verification_code;
```

```

// Essential headers for standard email sending
$headers = "From: no.reply.iparkthess@gmail.com\r\n";
$headers .= "Reply-To: no.reply.iparkthess@gmail.com\r\n";
$headers .= "Content-Type: text/plain; charset=UTF-8\r\n";

// Attempt to send the mail
if (mail($reipient_mail, $subject, $message, $headers)) {

} else {
    echo($reipient_mail);
    internal_error();
    exit;
}

return $verification_code;
}

```

3. Αποθήκευση και Λήξη: Ο κωδικός αποθηκεύεται στον πίνακα “otp” μαζί με τη χρονική σήμανση δημιουργίας του (“created_at”).

```

function send_otp($username) {
    if (is_user_verified($username)) {
        not_allowed();
    }

    $verification_code = sendmail($username);

    $user_id = get_user_id($username);

    $connection = connect();

    $update_stmt = "update otp set code = ? where user_id = ?";
    $insert_stmt = "insert into otp(code, user_id) VALUES (?, ?)";

    try {
        $queryString = "select * from otp where user_id = " . $user_id;
        $query_result = $connection->query($queryString);

        if ($query_result->num_rows > 0) {
            $stmt = $connection->prepare($update_stmt);

```

```

$row = $query_result->fetch_assoc();

date_default_timezone_set('Europe/Athens');
$otp_timestamp = strtotime($row['created_at']);
$ten_minutes_from_now = strtotime('+10 minutes');

if ($otp_timestamp <= $ten_minutes_from_now) {
    not_allowed();
}
} else {
    $stmt = $connection->prepare($insert_stmt);
}

$stmt->bind_param("ii", $verification_code, $user_id);
$stmt->execute();

header("HTTP/1.1 200 OK");
header("Content-Type: application/json");
echo '{"status": 200}';
exit;
} catch (mysqli_sql_exception $e) {
    echo $e;
    internal_error();
}
}

```

4. Επαλήθευση: Όταν ο χρήστης εισάγει τον κωδικό στην iOS εφαρμογή, ο server ελέγχει αν ο κωδικός είναι σωστός και αν η τρέχουσα ώρα δεν υπερβαίνει τα 10 λεπτά από τη δημιουργία του:

```

$otp_row = $otp_rows->fetch_assoc();
$otp = $otp_row['code'];
date_default_timezone_set('Europe/Athens');
$otp_timestamp = strtotime($otp_row['created_at']);
$ten_minutes_from_now = strtotime('+10 minutes');
// 10 minutes expiration check
if ($otp_timestamp > $ten_minutes_from_now) {
    echo $otp_timestamp;
    echo "\n";
    echo $ten_minutes_from_now;
    expired();
}

```

```
}
```

4.3 Αναλυτική Τεκμηρίωση των Endpoints

4.3.1 Endpoint: “/status.php” (Κατάσταση Διακομιστή)

Επιτρέπει στην εφαρμογή να ελέγχει αν ο server λειτουργεί κανονικά.

- Μέθοδος HTTP: “GET”
- Κεφαλίδες Request: None
- Παράμετροι: None
- JSON Απάντηση (Success 200 OK):

```
{  
  "status": 200  
}
```

4.3.2 Endpoint: “/user.php” (Διαχείριση Χρηστών)

Εγγραφή Χρήστη (Sign Up):

- Μέθοδος HTTP: “POST”
- Κεφαλίδες Request: “Authorization: Basic <Base64(Username:Password)>”
- JSON Body:

```
{  
  "email": "theocharisapo@gmail.com",  
  "name": "Θεοχάρης Αποστολίδης"  
}
```

- JSON Απάντηση (Success 200 OK):

```
{  
  "status": 200  
}
```

- JSON Απάντηση (Conflict 409 - Username Exists):

```
{
  "status": 409,
  "type": "USER_ALREADY_EXISTS"
}
```

Λήψη Πληροφοριών Χρήστη:

- Μέθοδος HTTP: “GET”
- Κεφαλίδες Request: “Authorization: Basic <Base64(Username:Password)>”
- JSON Απάντηση (Success 200 OK):

```
{
  "id": 1,
  "fullName": "Γιάννης Παπαδόπουλος",
  "username": "giannis",
  "email": "user@example.com",
  "ratings": [
    {
      "id": "1",
      "name": "PARKING KARAVAN SARAI",
      "rating": 5
    }
  ],
  "verified": true
}
```

4.3.3 Endpoint: “/verify.php” (Επαλήθευση OTP)

Αποστολή Κωδικού OTP:

- Μέθοδος HTTP: “GET”
- Κεφαλίδες Request: “Authorization: Basic <Base64(Username:Password)>”
- JSON Απάντηση (Success 200 OK):

```
{
  "status": 200
}
```

Έλεγχος Κωδικού OTP:

- Μέθοδος HTTP: “POST”
- Κεφαλίδες Request: “Authorization: Basic <Base64(Username:Password)>”
- Παράμετροι Query: “verification_code=123456”

- JSON Απάντηση (Success 200 OK):

```
{
  "status": 200
}
```

4.3.4 Endpoint: “/parkings.php” (Αναζήτηση Πάρκινγκ)

Επιστρέφει τη λίστα των χώρων στάθμευσης. Υποστηρίζει γεωγραφικό φιλτράρισμα βάσει ορίων χάρτη.

- Μέθοδος HTTP: “GET”
- Κεφαλίδες Request: “Authorization: Basic <Base64(Username:Password)>” (Προαιρετικό)
- Παράμετροι Query: “minLongitude”, “minLatitude”, “maxLongitude”, “maxLatitude”
- JSON Απάντηση (Success 200 OK):

```
[
  {
    "id": 1,
    "name": "PARKING KARAVAN SARAI",
    "latitude": 40.638031,
    "longitude": 22.942192,
    "address": "Ioustinianou 6, Thessaloniki 54631",
    "telephone": "2310272911",
    "max_car_spots": 250,
    "available_car_spots": 47,
    "max_motorbike_spots": 25,
    "available_motorbike_spots": 16,
    "amenities": {
      "handicap_spots_available": 7,
      "max_handicap_spots": 10,
      "electric_chargers_available": 4,
      "max_electric_chargers": 3,
      "car_wash": false,
      "valet": true,
      "gas_station": false
    },
    "schedule": {
      "monday": {
        "open": "07:00",
        "close": "23:00"
      }
    }
  },
]
```

```
"images": [  
  "/ptyxiaki/assets/1-1.png"  
],  
"rating": 4.5,  
"favourite": true  
}  
]
```

4.3.5 Endpoint: “/favourites.php” (Αγαπημένα)

Λήψη Αγαπημένων:

- Μέθοδος HTTP: “GET”
- Κεφαλίδες Request: “Authorization: Basic <Base64(Username:Password)>”
- JSON Απάντηση (Success 200 OK):

```
[  
  {  
    "id": 1,  
    "name": "PARKING KARAVAN SARAI",  
    "latitude": 40.638031,  
    "longitude": 22.942192,  
    "address": "Ioustinianou 6, Thessaloniki 54631",  
    "telephone": "2310272911",  
    "max_car_spots": 250,  
    "available_car_spots": 47,  
    "max_motorbike_spots": 25,  
    "available_motorbike_spots": 16,  
    "amenities": {  
      "handicap_spots_available": 7,  
      "max_handicap_spots": 10,  
      "electric_chargers_available": 4,  
      "max_electric_chargers": 3,  
      "car_wash": false,  
      "valet": true,  
      "gas_station": false  
    },  
    "schedule": {  
      "monday": {  
        "open": "07:00",  
        "close": "23:00"  
      }  
    },  
  },  
]
```

```
"images": [  
  "/ptyxiaki/assets/1-1.png"  
],  
"rating": 4.5,  
"favourite": true  
}  
]
```

Προσθήκη/Αφαίρεση Αγαπημένου (Toggle):

- Μέθοδος HTTP: “POST”
- Κεφαλίδες Request: “Authorization: Basic <Base64(Username:Password)>”
- Παράμετροι Query: “parking_id=1”
- JSON Απάντηση (Success 200 OK):

```
{  
  "status": 200  
}
```

4.3.6 Endpoint: “/rating.php” (Αξιολογήσεις)

Επιτρέπει στους χρήστες να βαθμολογούν (1-5) ή να ενημερώνουν τη βαθμολογία τους για ένα πάρκινγκ.

- Μέθοδος HTTP: “POST”
- Κεφαλίδες Request: “Authorization: Basic <Base64(Username:Password)>”
- JSON Body:

```
{  
  "parking_id": 1,  
  "rating": 4  
}
```

- JSON Απάντηση (Success 200 OK):

```
{  
  "status": 200  
}
```

4.3.7 Endpoint: “/change_password.php” (Αλλαγή Κωδικού)

- Μέθοδος HTTP: “POST”
- Κεφαλίδες Request: “Authorization: Basic <Base64(Username:Password)>”
- JSON Body:

```
{  
  "verification_code": 958241,  
  "password": "NewSecurePassword123"  
}
```

- JSON Απάντηση (Success 200 OK):

```
{  
  "status": 200  
}
```

4.3.8 Endpoint: “/forgot_password.php” (Ανάκτηση Κωδικού)

- Μέθοδος HTTP: “POST”
- Κεφαλίδες Request: None
- JSON Body:

```
{  
  "username": "giannis",  
  "verification_code": 958241,  
  "password": "NewSecurePassword123"  
}
```

- JSON Απάντηση (Success 200 OK):

```
{  
  "status": 200  
}
```

Κεφάλαιο 5ο: Τεχνολογίες Ανάπτυξης Εφαρμογής / Οικοσύστημα Apple

5.1 Εισαγωγή στο Οικοσύστημα της Apple

Η επιλογή της πλατφόρμας ανάπτυξης για μια σύγχρονη εφαρμογή εύρεσης θέσεων στάθμευσης αποτελεί κομβικό βήμα που καθορίζει την αρχιτεκτονική, την απόδοση, την ασφάλεια και τη γενικότερη εμπειρία του τελικού χρήστη. Για την υλοποίηση του iParkThess, επιλέχθηκε η ανάπτυξη μιας native εφαρμογής για το λειτουργικό σύστημα iOS της Apple.

Το οικοσύστημα ανάπτυξης της Apple χαρακτηρίζεται από μια στενά ολοκληρωμένη προσέγγιση, όπου το υλικό, το λειτουργικό σύστημα (software/OS) και τα εργαλεία ανάπτυξης (software development tools) σχεδιάζονται και βελτιστοποιούνται από την ίδια εταιρεία. Αυτή η στενή σύνδεση επιτρέπει στους προγραμματιστές να εκμεταλλεύονται στο έπακρο τις δυνατότητες της συσκευής, όπως τη μονάδα GPS, τις δυνατότητες τοπικής κρυπτογράφησης και τις τεχνολογίες γραφικών, προσφέροντας παράλληλα μια εξαιρετικά ομαλή και αποκρινόμενη εμπειρία χρήσης καθώς και την συνεργασία όλων των συσκευών της Apple μεταξύ τους, με προϋπόθεση να έχουν τον ίδιο iCloud λογαριασμό, όπως:

- Αντιγραφή σε μία συσκευή και επικόλληση σε μία άλλη
- Απάντηση τηλεφώνου / ανάγνωση μηνυμάτων από συσκευή χωρίς SIM κάρτα με χρήση της συσκευής που έχει SIM

Σε αντίθεση με τις cross-platform λύσεις (όπως το Flutter ή το React Native), η native ανάπτυξη σε περιβάλλον iOS εξασφαλίζει:

- Μέγιστη Απόδοση (Native Performance): Ο κώδικας μεταγλωττίζεται απευθείας σε assembly για την αρχιτεκτονική των επεξεργαστών της Apple (Apple Silicon / ARM), εξαλείφοντας ενδιάμεσα επίπεδα μετάφρασης (bridges).
- Άμεση Πρόσβαση σε Νέες Τεχνολογίες: Κάθε νέα δυνατότητα του iOS (π.χ. Live Activities, νέα APIs χαρτών) είναι άμεσα διαθέσιμη προς χρήση χωρίς την ανάγκη αναμονής για ενημερώσεις βιβλιοθηκών τρίτων.
- Συνέπεια στη Διεπαφή Χρήστη (UI Consistency): Τα στοιχεία διεπαφής ακολουθούν πιστά τις οδηγίες σχεδίασης της Apple (Human Interface Guidelines - HIG), προσφέροντας μια οικεία εμπειρία στον χρήστη.

5.1.1 Η Γλώσσα Προγραμματισμού Swift

Η Swift παρουσιάστηκε από την Apple το 2014 ως αντικαταστάτης της Objective-C. Πρόκειται για μια σύγχρονη γλώσσα προγραμματισμού γενικής χρήσης, η οποία σχεδιάστηκε με γνώμονα την ασφάλεια τύπων (type safety), την ταχύτητα (speed) και την εκφραστικότητα (expressiveness) [8].

Τα κυριότερα χαρακτηριστικά της Swift που αξιοποιήθηκαν στην εφαρμογή iParkThess περιλαμβάνουν:

- **Ασφάλεια Τύπων (Type Safety) και Έλεγχος Τύπων:** Η Swift είναι αυστηρά τυποποιημένη γλώσσα. Ο compiler ελέγχει τους τύπους των δεδομένων κατά τη μεταγλώττιση, αποτρέποντας σφάλματα ασυμβατότητας τύπων κατά το runtime.
- **Διαχείριση Προαιρετικών Τιμών (Optionals):** Μέσω του τύπου “Optional”, η Swift επιβάλλει τον ρητό χειρισμό περιπτώσεων όπου μια τιμή μπορεί να είναι απύσχα (“nil”). Αυτό μειώνει δραστικά τα runtime crashes που οφείλονται σε null-pointer εξαιρέσεις.
- **Αυτόματη Μέτρηση Αναφορών (Automatic Reference Counting - ARC):** Η μνήμη της εφαρμογής διαχειρίζεται αυτόματα μέσω του ARC, το οποίο αποδεσμεύει τη μνήμη που καταλαμβάνουν τα αντικείμενα (κλάσεις) όταν δεν υπάρχουν πλέον ενεργές αναφορές σε αυτά, αποφεύγοντας το overhead ενός garbage collector.
- **Value Types εναντίον Reference Types:** Η εκτενής χρήση δομών (“struct”) και απαριθμήσεων (“enum”) ως value types διασφαλίζει ότι τα δεδομένα αντιγράφονται κατά τη μεταβίβασή τους, αποτρέποντας ανεπιθύμητες παρενέργειες από κοινή χρήση μνήμης. Αμα ο προγραμματιστής θέλει να μεταβιβάζονται οι τιμές μπορεί πάντα να χρησιμοποιείσει και κλάσεις (“class”) οι οποίες είναι reference types και η πληροφορία παραμένει στην ίδια θέση μνήμης.

Στο παρακάτω κομμάτι κώδικα φαίνεται η διαφορά μεταξύ value types και reference types με την σύγκριση των ίδιων τιμών κώδικα Το print(a1.name) θα μας βγάλει σαν αποτέλεσμα “Theocharis” ενώ το print(b1.name) θα μας βγάλει “Nikos”. Αυτό συμβαίνει επειδή το a1 και το a2 είναι struct δεν είναι κοινό αντικείμενο, αλλά καινούργιο στην ανάθεση “var a2 = a1”. Αντίστοιχα στην ανάθεση “var b2 = b1” δεν δημιουργείται νέο αντικείμενο αλλά διατηρείται το παλιό οπότε όποια αλλαγή γίνεται στο b2 γίνεται αυτόματα και στο b1 γιατί στην πραγματικότητα είναι το ίδιο αντικείμενο.

```
struct A {  
    var name: String  
}  
  
class B {  
    var name: String
```

```
init(name: String) {  
    self.name = name  
}  
}  
  
var a1 = A(name: "Theocharis")  
var a2 = a1  
a2.name = "Nikos"  
print(a1.name)  
  
var b1 = B(name: "Theocharis")  
var b2 = b1  
b2.name = "Nikos"  
print(b1.name)
```

5.2 Το Ολοκληρωμένο Περιβάλλον Ανάπτυξης Xcode

Το Xcode αποτελεί το επίσημο και μοναδικό Ολοκληρωμένο Περιβάλλον Ανάπτυξης (Integrated Development Environment - IDE) που παρέχεται από την Apple για τη δημιουργία λογισμικού για τις πλατφόρμες macOS, iOS, iPadOS, watchOS και tvOS. Πρόκειται για ένα πανίσχυρο εργαλείο που ενοποιεί όλα τα στάδια της ανάπτυξης λογισμικού, από τη συγγραφή του κώδικα και τη σχεδίαση της διεπαφής, μέχρι τη μεταγλώττιση, την αποσφαλμάτωση (debugging), τη μέτρηση επιδόσεων (profiling) και την υποβολή της εφαρμογής στο App Store.

5.2.1 Βασικά Χαρακτηριστικά του Xcode

- **Επεξεργαστής Κώδικα (Source Editor):** Διαθέτει έξυπνη συμπλήρωση κώδικα (code completion μέσω του SourceKit), δυναμική επισήμανση συντακτικών σφαλμάτων, εργαλεία αυτόματης αναδιοργάνωσης κώδικα (refactoring) και ενσωμάτωση συστημάτων ελέγχου εκδόσεων (Git).
- **Μεταγλωττιστής LLVM (LLVM Compiler):** Ο μεταγλωττιστής LLVM αποτελεί την καρδιά του Xcode. Μετατρέπει τον κώδικα Swift σε βέλτιστο κώδικα μηχανής. Διακρίνεται για την ταχύτητά του και τα εξαιρετικά διαγνωστικά μηνύματα που βοηθούν στον εντοπισμό σφαλμάτων κατά τη φάση της μεταγλώττισης.
- **Xcode Simulator:** Επιτρέπει την προσομοίωση πολλαπλών συσκευών Apple (iPhone, iPad) με διαφορετικές εκδόσεις λειτουργικού συστήματος. Ο Simulator υποστηρίζει την προσομοίωση

γεωγραφικής θέσης (GPS mock locations), η οποία υπήρξε κρίσιμη για τη δοκιμή της λειτουργίας εύρεσης κοντινών πάρκινγκ στην εφαρμογή iParkThess.

- **Instruments:** Πρόκειται για ένα εξειδικευμένο εργαλείο ανάλυσης επιδόσεων. Επιτρέπει στους προγραμματιστές να παρακολουθούν τη χρήση της μνήμης (εντοπισμός memory leaks), την κατανάλωση της CPU, τη θερμική επιβάρυνση της συσκευής και τη χρήση του δικτύου σε πραγματικό χρόνο.
- **Asset Catalogs:** Μέσω αυτών γίνεται η διαχείριση όλων των οπτικών πόρων της εφαρμογής (εικόνες, εικονίδια, χρώματα). Υποστηρίζουν την αυτόματη προσαρμογή των πόρων ανάλογα με την ανάλυση της οθόνης (@1x @2x, @3x) και τη λειτουργία εμφάνισης (Light/Dark Mode).

5.3 Απαιτήσεις και Περιορισμοί Ανάπτυξης σε Περιβάλλον Apple

Ένα από τα πιο χαρακτηριστικά γνωρίσματα της ανάπτυξης εφαρμογών για το iOS είναι η αυστηρή δέσμευση στο hardware και software της Apple. Η συγγραφή, η μεταγλώττιση, η δοκιμή σε φυσικές συσκευές και η διανομή μιας εφαρμογής iOS απαιτούν αποκλειστικά τη χρήση υπολογιστή Mac ή Macbook που εκτελεί το λειτουργικό σύστημα macOS.

5.3.1 Γιατί είναι απαραίτητο το macOS;

Ο περιορισμός αυτός δεν αποτελεί απλώς μια εμπορική απόφαση της Apple, αλλά βασίζεται σε βαθιές τεχνικές παραμέτρους:

- **Μεταγλώττιση και Build Tools:** Η εργαλειοθήκη μεταγλώττισης (Xcode Command Line Tools, compiler, linker) είναι σχεδιασμένη και βελτιστοποιημένη αποκλειστικά για το macOS. Δεν υφίσταται επίσημη έκδοση του Xcode για λειτουργικά συστήματα Windows ή Linux.
- **Υπογραφή Κώδικα (Code Signing):** Η Apple επιβάλλει αυστηρούς κανόνες ασφαλείας για την εκτέλεση λογισμικού στις συσκευές της. Κάθε εφαρμογή πρέπει να υπογραφεί ψηφιακά με πιστοποιητικά που εκδίδονται από την Apple. Η διαδικασία αυτή ελέγχεται μέσω των keychain services και των εργαλείων ασφαλείας που είναι ενσωματωμένα στο macOS.
- **Προσομοίωση και Διασύνδεση:** Ο Xcode Simulator βασίζεται σε εσωτερικά frameworks του macOS για την προσομοίωση του iOS kernel και του γραφικού περιβάλλοντος, κάτι που καθιστά αδύνατη τη μεταφορά του σε άλλα λειτουργικά συστήματα χωρίς virtualization, το οποίο όμως παραβιάζει τους όρους χρήσης (EULA) της Apple.
- **Διανομή μέσω App Store:** Η υποβολή της εφαρμογής στην πλατφόρμα App Store Connect γίνεται μέσω του Xcode ή της εφαρμογής Transporter, εργαλεία που διατίθενται αποκλειστικά για macOS.

Για την ανάπτυξη του iParkThess, η χρήση ενός Macbook επέτρεψε την απρόσκοπτη μεταγλώττιση του κώδικα Swift, τη δοκιμή της εφαρμογής σε πραγματικό χρόνο στον Simulator και τη διαχείριση των απαραίτητων provisioning profiles για τη δοκιμή της εφαρμογής σε φυσική συσκευή iPhone με την χρήση automatic signing δίνοντας τη δυνατότητα δοκιμής της εφαρμογής σε φυσική συσκευή χωρίς να χρειάζεται Apple Developer Account το οποίο είναι πρόγραμμα με ετήσια συνδρομή.

5.4 Σύγκριση Τεχνολογιών: UIKit εναντίον SwiftUI

Κατά τον σχεδιασμό της εφαρμογής iParkThess, εξετάστηκαν οι δύο βασικές τεχνολογίες που προσφέρει η Apple για τη δημιουργία διεπαφών χρήστη: το παραδοσιακό UIKit και το σύγχρονο SwiftUI.

5.4.1 UIKit: Το Παραδοσιακό Framework

Το UIKit αποτελεί τη βάση για την ανάπτυξη iOS εφαρμογών από την πρώτη έκδοση του iPhone SDK το 2008. Βασίζεται στο αρχιτεκτονικό πρότυπο Model-View-Controller (MVC) και ακολουθεί την προσέγγιση του Προστακτικού Προγραμματισμού (Imperative Programming).

Στον προστακτικό προγραμματισμό, ο προγραμματιστής πρέπει να περιγράψει αναλυτικά πώς θα γίνουν οι αλλαγές στη διεπαφή. Για παράδειγμα, αν μια εφαρμογή πρέπει να εμφανίσει ένα μήνυμα σφάλματος όταν αποτύχει μια κλήση δικτύου, ο κώδικας πρέπει να [9]:

- Ελέγξει την αποτυχία.
- Δημιουργήσει ένα αντικείμενο “UILabel” ή “UIAlertController”.
- Ορίσει τις ιδιότητές του (κείμενο, χρώμα, γραμματοσειρά).
- Προσθέσει το στοιχείο στην ιεραρχία των views (“addSubview”).
- Ορίσει τους γεωγραφικούς περιορισμούς (Auto Layout constraints).
- Διαχειριστεί την αφαίρεση ή την απόκρυψή του όταν το σφάλμα επιλυθεί.

Αυτή η προσέγγιση, αν και εξαιρετικά ευέλικτη και δοκιμασμένη στο χρόνο, οδηγεί συχνά σε:

- Μεγάλο όγκο κώδικα (boilerplate code).

- Αυξημένο κίνδυνο ασυνέπειας μεταξύ της κατάστασης της εφαρμογής (state) και της διεπαφής (UI out-of-sync).
- Πολύπλοκη διαχείριση των Auto Layout constraints, ειδικά σε δυναμικές οθόνες.

5.4.2 SwiftUI: Το Σύγχρονο Δηλωτικό Framework

Το SwiftUI παρουσιάστηκε από την Apple το 2019 και αντιπροσωπεύει μια ριζική αλλαγή παραδείγματος (paradigm shift). Αποτελεί ένα Δηλωτικό Framework (Declarative Framework), όπου ο προγραμματιστής περιγράφει **τι** πρέπει να εμφανίζεται η διεπαφή με βάση την τρέχουσα κατάσταση (state) της εφαρμογής, και όχι το **πώς** θα γίνει η μετάβαση σε αυτή την κατάσταση.

5.4.3 Σύγκριση Αρχιτεκτονικής UI

Στο UIKit, η σχεδίαση γίνεται συχνά μέσω Storyboards ή αρχείων XIB (XML μορφή), τα οποία είναι δύσκολο να διαχειριστούν σε περιβάλλοντα ομάδων (merge conflicts στο Git) και δεν παρέχουν εύκολη επαναχρησιμοποίηση στοιχείων. Εναλλακτικά, η προγραμματιστική σχεδίαση απαιτεί εκατοντάδες γραμμές Auto Layout κώδικα.

Αντίθετα, στο SwiftUI, η σχεδίαση γίνεται αποκλειστικά σε Swift κώδικα. Η διάταξη ορίζεται χρησιμοποιώντας Stacks (VStack για κατακόρυφη, HStack για οριζόντια, ZStack για επικάλυψη), καθιστώντας τον κώδικα εξαιρετικά ευανάγνωστο και καθαρό:

```
VStack(alignment: .leading, spacing: 8) {
    Text("Όνομα Πάρκινγκ")
        .font(.headline)
        .foregroundColor(.primary)
    Text("Διαθέσιμες Θέσεις: 45")
        .font(.subheadline)
        .foregroundColor(.green)
}
.padding()
.background(Color(.systemBackground))
.cornerRadius(12)
```

5.5 Γιατί η SwiftUI Επιλέχθηκε για το Έργο iParkThess

Η απόφαση για τη χρήση της SwiftUI ως αποκλειστικής τεχνολογίας για τη δημιουργία της διεπαφής του iParkThess βασίστηκε σε μια σειρά από στρατηγικά και τεχνικά πλεονεκτήματα:

5.5.1 Στρατηγική Επιλογή της Apple (Future-Proofing)

Η Apple έχει καταστήσει σαφές ότι η SwiftUI αποτελεί το μέλλον της ανάπτυξης εφαρμογών σε όλες τις πλατφόρμες της. Τα νέα APIs και τα καινούργια frameworks (όπως το MapKit για SwiftUI που κυκλοφόρησε πρόσφατα, ή τα Widgets) σχεδιάζονται με προτεραιότητα στη SwiftUI. Επιλέγοντας τη SwiftUI, διασφαλίζεται ότι η εφαρμογή iParkThess θα παραμείνει συμβατή και εύκολα συντηρήσιμη για τα επόμενα χρόνια.

5.5.2 Ταχύτητα Ανάπτυξης (Rapid Prototyping)

Η δηλωτική σύνταξη της SwiftUI επιτρέπει τη δημιουργία πολύπλοκων διεπαφών με ένα κλάσμα του κώδικα που θα απαιτούνταν στο UIKit. Για παράδειγμα, η δημιουργία μιας λίστας στο UIKit απαιτεί την υλοποίηση των protocols “UITableViewDataSource” και “UITableViewDelegate”, τη δημιουργία custom “UITableViewCell” και τη χειροκίνητη εγγραφή των Cells. Στη SwiftUI, η ίδια λίστα υλοποιείται με ελάχιστες γραμμές:

```
List(viewModel.parkings) { parking in
    ParkingRow(parking: parking)
}
```

Αυτή η απλότητα επέτρεψε τη γρήγορη υλοποίηση και τροποποίηση των οθονών της εφαρμογής κατά τη διάρκεια των δοκιμών.

5.5.3 Previews και Real-time Debugging

Ένα από τα πιο δυνατά χαρακτηριστικά της SwiftUI στο Xcode είναι η λειτουργία SwiftUI Previews. Επιτρέπει στον προγραμματιστή να βλέπει τη διεπαφή που αναπτύσσει σε πραγματικό χρόνο, ακριβώς δίπλα από τον κώδικα, χωρίς να χρειάζεται να μεταγλωττίσει ολόκληρη την εφαρμογή και να την εκτελέσει στον Simulator ή σε φυσική συσκευή.

Τα previews υποστηρίζουν:

- **Interactive Mode:** Δυνατότητα αλληλεπίδρασης με το view (κλικ, σκρολάρισμα, εισαγωγή κειμένου) σαν να εκτελείται σε πραγματική συσκευή.
- **Multi-device Preview:** Ταυτόχρονη προβολή της ίδιας οθόνης σε διαφορετικά μοντέλα iPhone, με διαφορετικό προσανατολισμό (Portrait/Landscape) ή με ενεργοποιημένο το Dark Mode.
- **Localization Previews:** Προβολή της οθόνης σε διαφορετικές γλώσσες (Ελληνικά/Αγγλικά) για τον άμεσο έλεγχο της σωστής στοίχισης των κειμένων.

Αυτό μείωσε δραστικά τον χρόνο του debugging, καθώς οι αλλαγές στο UI εμφανίζονταν ακαριαία στην οθόνη του Xcode.

5.5.4 Εγγενείς Βελτιστοποιήσεις και Απόδοση

Τα views στη SwiftUI δεν είναι κλάσεις (classes) που κληρονομούν από το “UIView” όπως στο UIKit, αλλά δομές δεδομένων (value type structs). Αυτό σημαίνει ότι η δημιουργία και η καταστροφή τους είναι εξαιρετικά ελαφριές διεργασίες που δεν επιβαρύνουν τη μνήμη της συσκευής (garbage collection / reference counting overhead). Το SwiftUI επανασχεδιάζει τη διεπαφή μόνο όταν υπάρχει πραγματική αλλαγή στα δεδομένα, εξασφαλίζοντας 60 ή 120 καρέ το δευτερόλεπτο (FPS) στις μεταβάσεις και τα animations, κάτι που είναι κρίσιμο κατά τη μετακίνηση του χάρτη στο iParkThess.

5.5.5 Διαχείριση Ροής Δεδομένων (Property Wrappers)

Η SwiftUI ενσωματώνει Property Wrappers που απλοποιούν τη ροή των δεδομένων (data flow) μεταξύ του επιχειρησιακού κώδικα (Business Logic) και της διεπαφής χρήστη. Αυτό υποστηρίζει άμεσα το αρχιτεκτονικό πρότυπο MVVM [10]:

- “@State”: Χρησιμοποιείται για την αποθήκευση τοπικής κατάστασης που ανήκει αποκλειστικά σε ένα view (π.χ. αν ένα μενού φίλτρων είναι ανοιχτό ή κλειστό).
- “@Binding”: Δημιουργεί μια αμφίδρομη σύνδεση (two-way binding) μεταξύ ενός view και μιας κατάστασης που ανήκει σε ένα γονικό view. Χρησιμοποιείται ευρέως σε custom components (π.χ. custom TextFields).
- “@Published”: Τοποθετείται μέσα στα ViewModels. Δηλώνει ότι οποιαδήποτε αλλαγή σε αυτή την ιδιότητα θα πρέπει να προκαλέσει την ενημέρωση των views που την παρακολουθούν.
- “@StateObject”: Χρησιμοποιείται για την αρχικοποίηση ενός ViewModel μέσα σε ένα view. Εξασφαλίζει ότι το ViewModel δεν θα καταστραφεί και δεν θα ξαναδημιουργηθεί κατά το επανασχεδιασμό του view.
- “@EnvironmentObject”: Επιτρέπει τη διάχυση δεδομένων σε ολόκληρη την ιεραρχία των views χωρίς την ανάγκη χειροκίνητης μεταβίβασης των παραμέτρων (dependency injection). Στο iParkThess, χρησιμοποιείται για τη διανομή του “LocationManager” και του “LoaderManager”.

Η χρήση αυτών των εργαλείων επέτρεψε τη δημιουργία μιας καθαρής αρχιτεκτονικής, όπου ο κώδικας του UI παραμένει απαλλαγμένος από διαχειριστικές λεπτομέρειες των δεδομένων, εστιάζοντας αποκλειστικά στην παρουσίασή τους.

5.5.6 Διαχείριση Μνήμης και Lifecycle

Σε αντίθεση με το UIKit, όπου ο κύκλος ζωής ελέγχεται από μεθόδους όπως το “viewDidLoad”, “viewWillAppear” και “viewWillDisappear” σε View Controllers, η SwiftUI εισάγει μια απλοποιημένη προσέγγιση βασισμένη σε γεγονότα [11]:

- “onAppear”: Εκτελείται όταν το view εμφανίζεται στην ιεραρχία των οθονών. Στο iParkThess, χρησιμοποιείται για την εκκίνηση φόρτωσης δεδομένων ή τη σύνδεση με το Location Manager.
- “onDisappear”: Εκτελείται όταν το view αφαιρείται από την οθόνη. Χρησιμοποιείται για την ακύρωση εκκρεμών διεργασιών ή δικτυακών αιτημάτων.
- “onChange”: Παρακολουθεί μια τιμή κατάστασης (State) και εκτελεί κώδικα όταν αυτή αλλάξει (π.χ. ενημέρωση του loader manager όταν μεταβάλλεται η κατάσταση φόρτωσης).

Αυτή η προσέγγιση μειώνει δραστικά τις διαρροές μνήμης (memory leaks) και τα retain cycles, καθώς τα views ως structs δεν δημιουργούν ισχυρούς δεσμούς αναφοράς (strong references).

Κεφάλαιο 6ο: Ανάλυση, Αρχιτεκτονική και Οθόνες της Εφαρμογής

6.1 Αρχιτεκτονική της Εφαρμογής (Clean Architecture & MVVM)

Η εφαρμογή iParkThess σχεδιάστηκε και αναπτύχθηκε ακολουθώντας τις αρχές της Καθαρής Αρχιτεκτονικής (Clean Architecture) [12] σε συνδυασμό με το πρότυπο σχεδίασης Model-View-ViewModel (MVVM) για το επίπεδο παρουσίασης (Presentation Layer). Η επιλογή αυτή έγινε με σκοπό τη δημιουργία ενός συστήματος με χαμηλή σύζευξη (loose coupling), υψηλή συνεκτικότητα (high cohesion) και ευκολία στη συντήρηση και τη δοκιμή (testability) των επιμέρους τμημάτων του κώδικα.

6.1.1 Τα Επίπεδα του Clean Architecture

Η αρχιτεκτονική δομή της εφαρμογής iParkThess χωρίζεται σε τρία διακριτά επίπεδα (Layers), καθένα από τα οποία έχει συγκεκριμένες ευθύνες:

Επίπεδο Παρουσίασης (Presentation Layer):

- Views (SwiftUI Views): Είναι υπεύθυνα αποκλειστικά για τη σχεδίαση και την εμφάνιση της διεπαφής χρήστη. Είναι τελείως παθητικά (dumb views) και δεν περιέχουν λογική απόφασης. Αντιδρούν στις αλλαγές των δεδομένων που εκθέτει το ViewModel.
- ViewModels (ObservableObjects): Διαχειρίζονται την κατάσταση (state) της οθόνης. Λαμβάνουν τα γεγονότα χρήστη (user actions) από το View, καλούν τα κατάλληλα Use Cases από το επίπεδο Domain, και εκθέτουν τα αποτελέσματα πίσω στο View μέσω “@Published” ιδιοτήτων.

Επίπεδο Επιχειρησιακής Λογικής (Domain Layer):

- Είναι ο πυρήνας της εφαρμογής και παραμένει ανεξάρτητο από εξωτερικές βιβλιοθήκες, βάσεις δεδομένων ή το UI framework.
- Models: Αντιπροσωπεύουν τις βασικές οντότητες της εφαρμογής (π.χ. “Parking”, “User”, “Review”).
- Use Cases (Interactors): Περιέχουν τους κανόνες επιχειρησιακής λογικής (business rules) για μια συγκεκριμένη λειτουργία (π.χ. “GetNearestParkingsUseCase”, “ToggleFavouriteUseCase”, “SubmitRatingUseCase”).
- Repository Protocols (Interfaces): Ορίζουν τα συμβόλαια (συμφωνίες) για την άντληση και αποθήκευση δεδομένων, χωρίς να γνωρίζουν πώς ακριβώς υλοποιούνται αυτές οι λειτουργίες.

Επίπεδο Δεδομένων (Data Layer):

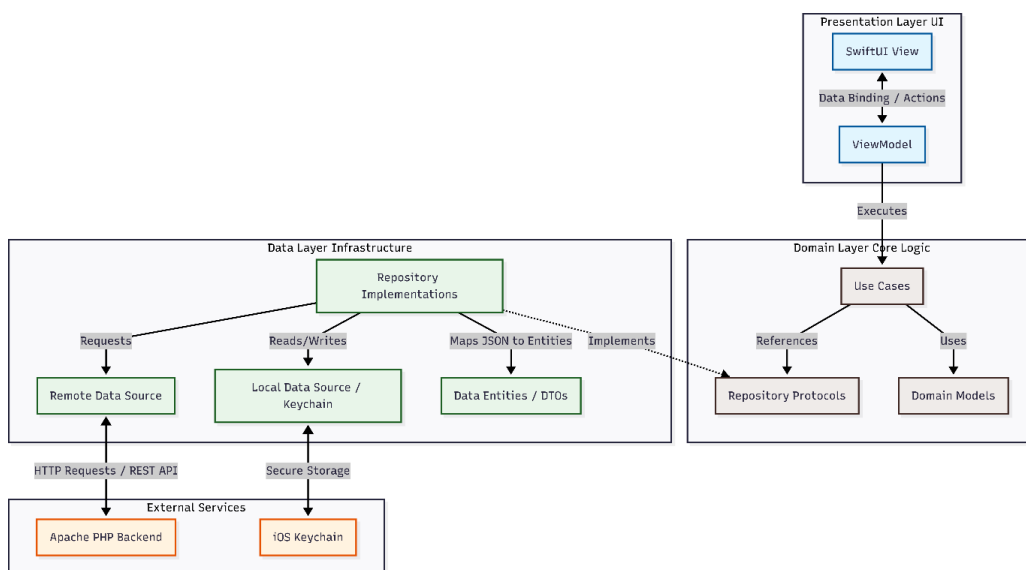
- **Repositories:** Υλοποιούν τα πρωτόκολλα (protocols) που ορίστηκαν στο Domain Layer. Αποφασίζουν αν θα αντλήσουν δεδομένα από τοπική μνήμη (Local Cache/Keychain) ή από το δίκτυο (Remote Server).
- **Data Sources (Remote & Local):** Πραγματοποιούν τις πραγματικές λειτουργίες ανάκτησης δεδομένων. Η “RemoteDataSource” εκτελεί HTTP αιτήματα API προς το PHP backend, ενώ η “LocalDataSource” αλληλεπιδρά με το Keychain ή το UserDefaults.
- **Entities (DTOs - Data Transfer Objects):** Αντιπροσωπεύουν τη μορφή με την οποία τα δεδομένα έρχονται από το API (JSON mapping).

6.1.2 Dependency Injection (Εγχυση Εξαρτήσεων)

Για τη διαχείριση των dependencies μεταξύ των επιπέδων, η εφαρμογή χρησιμοποιεί ένα σύστημα Dependency Injection (DI) μέσω της κλάσης “Injection” στο Core. Αυτό επιτρέπει στα ViewModels να λαμβάνουν τα Use Cases που χρειάζονται κατά την αρχικοποίησή τους, χωρίς να γνωρίζουν τις συγκεκριμένες κλάσεις υλοποίησης των Repositories ή των Data Sources, διευκολύνοντας την αντικατάστασή τους με mock αντικείμενα κατά τη συγγραφή ελέγχων (unit tests).

6.2 Αρχιτεκτονικό Διάγραμμα Εφαρμογής

Το παρακάτω διάγραμμα (Sequence/Data Flow Diagram) αναπαριστά γραφικά τη δομή της Clean Architecture και τον τρόπο με τον οποίο ρέουν τα δεδομένα από το UI προς το Backend API και αντίστροφα:



Εικόνα 6.1 Sequence Diagram της Clean Architecture

6.3 Περιγραφή Οθονών και Λειτουργικότητας

Στην ενότητα αυτή αναλύονται διεξοδικά όλες οι οθόνες της εφαρμογής iParkThess, η γραφική τους διεπαφή, τα ViewModels που τις υποστηρίζουν και οι επιχειρησιακές λειτουργίες που εκτελούν.

6.3.1 Splash View & Onboarding (SplashView / LobbyView)

Η οθόνη SplashView είναι η πρώτη οθόνη που εμφανίζεται κατά την εκκίνηση της εφαρμογής.

- Σκοπός: Επιτελεί διπλό ρόλο: αφενός προσφέρει μια ομαλή οπτικά εισαγωγή στον χρήστη εμφανίζοντας το λογότυπο της εφαρμογής με εφέ κλιμάκωσης (scale & opacity animation), και αφετέρου εκτελεί τους απαραίτητους αρχικούς ελέγχους στο παρασκήνιο.
- Λειτουργία: Κατά τη διάρκεια της προβολής της (περίπου 2 δευτερόλεπτα), το “AppViewModel” ελέγχει αν υπάρχουν αποθηκευμένα διαπιστευτήρια χρήστη στο iOS Keychain. Αν βρεθούν έγκυρα διαπιστευτήρια, η εφαρμογή πραγματοποιεί αυτόματα σύνδεση (auto-login) και ενημερώνει το “loginStatus” σε “.loggedIn”. Αν δεν βρεθούν, η εφαρμογή μεταβαίνει σε guest mode.
- LobbyView: Αποτελεί τον κεντρικό διαχειριστή πλοήγησης της εφαρμογής. Περιέχει ένα προσαρμοσμένο “TabView” με τέσσερις βασικές επιλογές (Αρχική, Χάρτης, Αναζήτηση, Λογαριασμός). Το “LobbyView” χρησιμοποιεί ένα “thinMaterial” εφέ για την κάτω μπάρα επιλογών (Tab Bar), επιτρέποντας στον χάρτη να φαίνεται διακριτικά από πίσω.



Εικόνα 6.2 Splash οθόνη (Dark Theme)



Εικόνα 6.3 Splash οθόνη (Light Theme)

6.3.2 Αρχική Οθόνη (HomeView)

Η οθόνη HomeView λειτουργεί ως ο προσωπικός πίνακας ελέγχου (dashboard) του χρήστη.

* Διασύνδεση & UI: Περιλαμβάνει μια δυναμική κεφαλίδα καλωσορίσματος που προσαρμόζεται ανάλογα με την ώρα της ημέρας (π.χ. "Καλημέρα, Θεοχάρη") και εμφανίζει το όνομα του συνδεδεμένου χρήστη. Η οθόνη υποστηρίζει pull-to-refresh, επιτρέποντας στον χρήστη να ανανεώσει άμεσα τα δεδομένα των πάρκινγκ σέρνοντας την οθόνη προς τα κάτω.

Λειτουργικότητα ανάλογα με το Auth State:

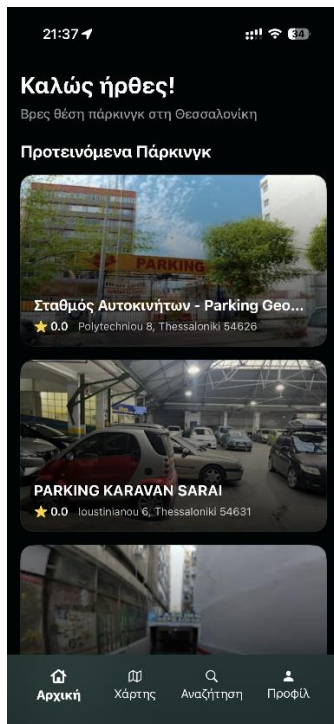
Συνδεδεμένος Χρήστης (Logged In):

- Αγαπημένα (Favourites Section): Εμφανίζει σε οριζόντιο scroll view τα πάρκινγκ που έχει αποθηκεύσει ο χρήστης ως αγαπημένα, δείχνοντας την τρέχουσα διαθεσιμότητά τους σε πραγματικό χρόνο. Αν δεν υπάρχουν αγαπημένα, εμφανίζεται ένα empty state με προτροπή για προσθήκη.
- Κοντινότερα (Nearest Section): Εμφανίζει μια λίστα με τους πλησιέστερους χώρους στάθμευσης, ταξινομημένους βάσει της γεωγραφικής απόστασης από την τρέχουσα θέση του χρήστη (λαμβάνεται μέσω του "LocationManager").

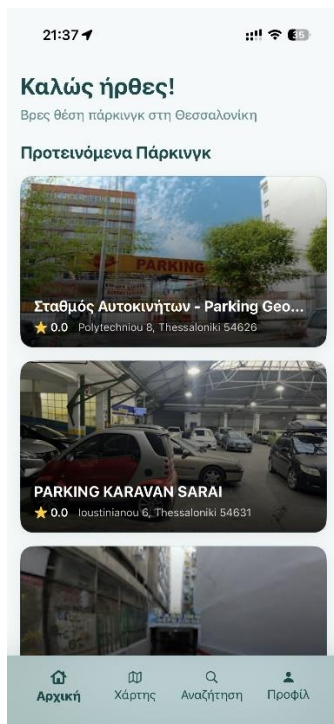
Επισκέπτης (Guest Mode):

- Εμφανίζει μια ενότητα με προτεινόμενα (Featured) πάρκινγκ της Θεσσαλονίκης.

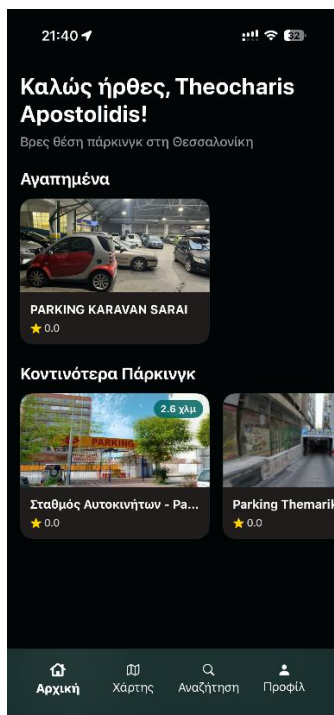
- Εμφανίζει προτροπές (call-to-action cards) που ενθαρρύνουν τον χρήστη να εγγραφεί ή να συνδεθεί για να ξεκλειδώσει δυνατότητες όπως τα αγαπημένα και η βαθμολόγηση.
- Αλληλεπίδραση: Tapping σε οποιοδήποτε πάρκινγκ ανοίγει ένα bottom sheet (“ParkingDetailsHalfView”) που παρουσιάζει τις λεπτομέρειές του.



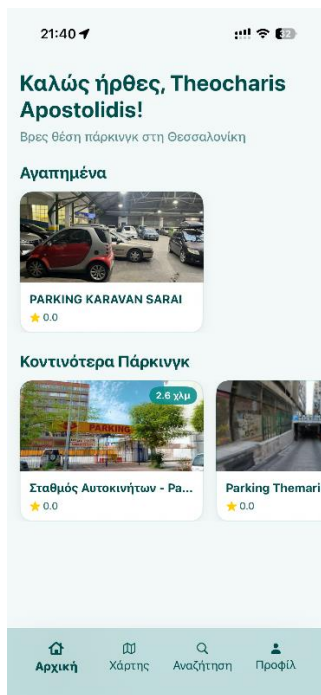
Εικόνα 6.4 Αρχική οθόνη αποσυνδεδεμένου χρήστη (Dark Theme)



Εικόνα 6.5 Αρχική οθόνη αποσυνδεδεμένου χρήστη (Light Theme)



Εικόνα 6.6 Αρχική οθόνη συνδεδεμένου χρήστη (Dark Theme)

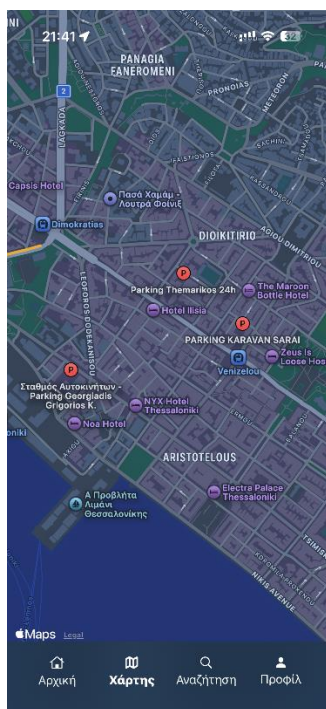


Εικόνα 6.7 Αρχική οθόνη συνδεδεμένου χρήστη (Light Theme)

6.3.3 Οθόνη Χάρτη (MapView)

Η οθόνη MapView αποτελεί τον πυρήνα της αλληλεπίδρασης της εφαρμογής iParkThess.

- Ενσωμάτωση Χάρτη: Χρησιμοποιεί το native framework “MapKit” της Apple, προσαρμοσμένο για SwiftUI. Εμφανίζει την τρέχουσα τοποθεσία του χρήστη με μια μπλε κουκκίδα και τοποθετεί καρφίτσες (Annotations) για κάθε χώρο στάθμευσης.



Εικόνα 6.8 Οθόνη Χάρτη (Dark Theme)



Εικόνα 6.9 Οθόνη Χάρτη (Light Theme)

Η εφαρμογή iParkThess ενσωματώνει μια προηγμένη βελτιστοποίηση για την ανάκτηση των πάρκινγκ. Αντί να καλεί όλα τα πάρκινγκ της πόλης από το backend, το “MapViewModel” παρακολουθεί τις αλλαγές στην κάμερα του χάρτη (“onMapCameraChange”) και υπολογίζει το ορατό γεωγραφικό παραλληλόγραμμο (Bounding Box) χρησιμοποιώντας τις μέγιστες και ελάχιστες τιμές γεωγραφικού πλάτους και μήκους.

Ο κώδικας υπολογισμού του Bounding Box στο “MapViewModel” είναι ο εξής:

```
func getVisibleCoordinatesBox(context: MapCameraUpdateContext) -> CoordinatesBox? {
    let region = context.region
    let center = region.center
    let span = region.span

    // Υπολογισμός ορίων με ανοχή (padding) 0.005 μοιρών
    let minLatitude = max(-90.0, center.latitude - span.latitudeDelta / 2 - 0.005)
    let maxLatitude = min(90.0, center.latitude + span.latitudeDelta / 2 + 0.005)
    let minLongitude = normalizeLongitude(center.longitude - span.longitudeDelta / 2 - 0.005)
    let maxLongitude = normalizeLongitude(center.longitude + span.longitudeDelta / 2 + 0.005)

    guard minLatitude < maxLatitude else { return nil }

    return CoordinatesBox(
        maxLatitude: maxLatitude,
        maxLongitude: maxLongitude,
```

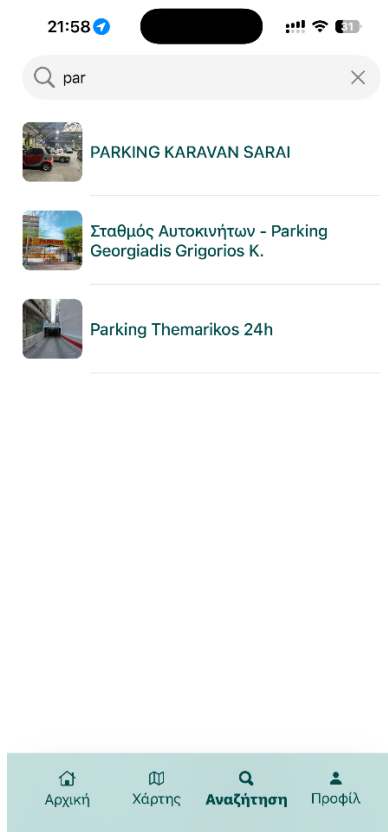
```
        minLatitude: minLatitude,  
        minLongitude: minLongitude  
    )  
}
```

Όταν η κάμερα σταθεροποιηθεί, εκτελείται ένα ασύγχρονο αίτημα δικτύου (“getParkings”) μεταφέροντας το Bounding Box ως παραμέτρους. Για την αποφυγή συνεχόμενων κλήσεων κατά το σύρσιμο (spamming), εφαρμόζεται τεχνική Debouncing μέσω ενός “DispatchWorkItem” που καθυστερεί την εκτέλεση κατά 100ms.

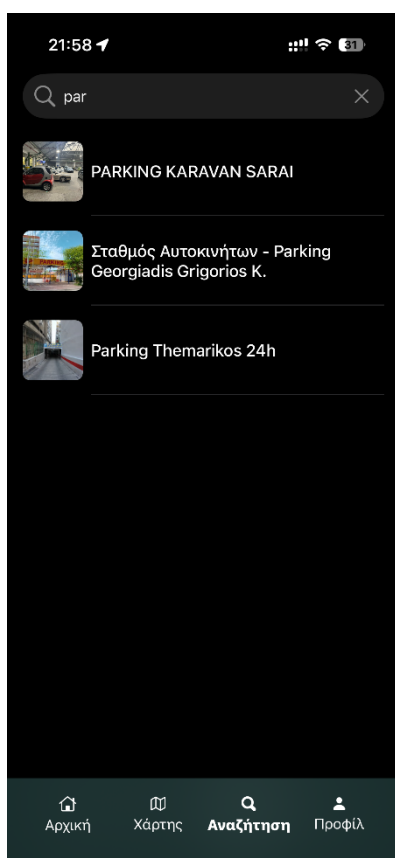
6.3.4 Αναζήτηση και Λίστα (SearchView)

Η οθόνη SearchView προσφέρει μια εναλλακτική, πιο δομημένη προσέγγιση για την εύρεση πάρκινγκ.

- Σκοπός: Επιτρέπει στον χρήστη να αναζητήσει χώρους στάθμευσης εισάγοντας κείμενο (όνομα πάρκινγκ ή διεύθυνση).
- Δυναμικό Φιλτράρισμα: Καθώς ο χρήστης πληκτρολογεί, η λίστα φιλτράρεται αυτόματα σε πραγματικό χρόνο. Κάθε στοιχείο της λίστας εμφανίζει το όνομα του πάρκινγκ, τη διεύθυνσή του, τη γεωγραφική απόσταση από τον χρήστη, την τρέχουσα βαθμολογία του (σε αστέρια) και την κατάσταση διαθεσιμότητας (διαθέσιμες/συνολικές θέσεις).
- Μετάβαση: Ο χρήστης μπορεί να επιλέξει ένα πάρκινγκ για να δει τις λεπτομέρειές του, ή να πατήσει το εικονίδιο του χάρτη δίπλα στο όνομα για να μεταφερθεί αυτόματα στην οθόνη του χάρτη με εστίαση στο συγκεκριμένο σημείο.



Εικόνα 6.10 Οθόνη Αναζήτησης (Light Theme)



Εικόνα 6.11 Οθόνη Αναζήτησης (Dark Theme)

6.3.5 Πληροφορίες Χώρου Στάθμευσης (ParkingDetailsView / ParkingDetailsHalfView)

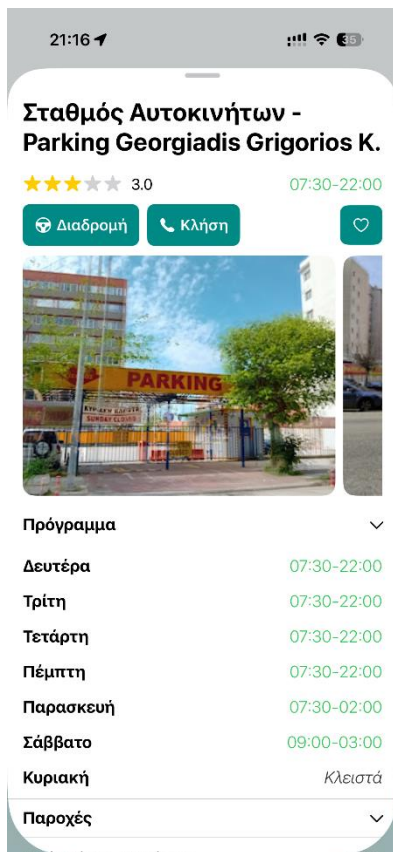
Αυτή η οθόνη παρέχει όλες τις διαθέσιμες πληροφορίες για έναν επιλεγμένο χώρο στάθμευσης.

Δομή Πληροφοριών:

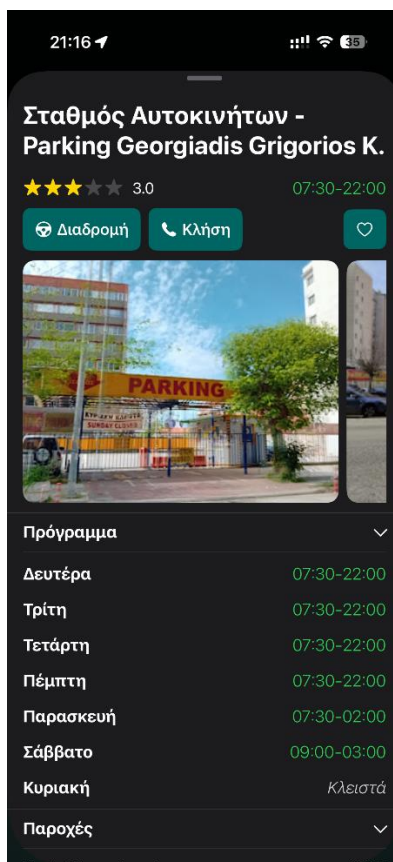
- Κεφαλίδα με Φωτογραφίες: Carousel εικόνων του πάρκινγκ.
- Βασικές Πληροφορίες: Όνομα, διεύθυνση, τηλέφωνο επικοινωνίας (με δυνατότητα άμεσης κλήσης).
- Δείκτης Live Διαθεσιμότητας: Ένα progress ring που δείχνει το ποσοστό των ελεύθερων θέσεων.
- Ωράριο Λειτουργίας: Αναλυτικός πίνακας με τις ώρες λειτουργίας ανά ημέρα της εβδομάδας, με ένδειξη αν το πάρκινγκ είναι ανοιχτό αυτή τη στιγμή.
- Παροχές (Amenities List): Εικονίδια για όλες τις παρεχόμενες υπηρεσίες.
- Αξιολογήσεις Χρηστών: Προβολή του μέσου όρου βαθμολογίας και των γραπτών σχολίων άλλων χρηστών.

Ενέργειες Χρήστη:

- Αγαπημένο: Κουμπί σε σχήμα καρδιάς που προσθέτει/αφαιρεί το πάρκινγκ από τα αγαπημένα του χρήστη (εκτελεί API κλήση στο “/favourites.php” και ενημερώνει το τοπικό state).
- Οδηγίες Πλοήγησης (Directions): Ανοίγει ένα Action Sheet επιτρέποντας στον χρήστη να επιλέξει την εφαρμογή πλοήγησης της προτίμησής του (Apple Maps ή Google Maps), μεταφέροντας τις συντεταγμένες προορισμού.
- Αξιολόγηση: Κουμπί που μεταφέρει τον χρήστη στην οθόνη υποβολής βαθμολογίας (RateView).



Εικόνα 6.12 Οθόνη Πληροφοριών παρκινγκ (Light Theme)

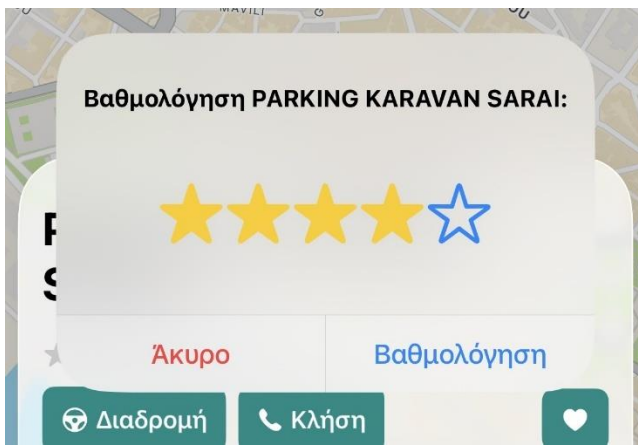


Εικόνα 6.13 Οθόνη Πληροφοριών παρκινγκ (Dark Theme)

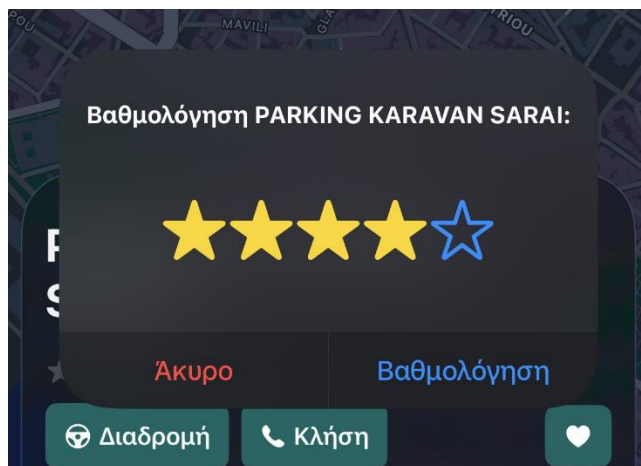
6.3.6 Σύστημα Αξιολόγησης (RateView)

Η οθόνη RateView επιτρέπει τη διαδραστική βαθμολόγηση ενός πάρκινγκ.

- UI/UX: Εμφανίζει 5 αστέρια. Ο χρήστης μπορεί να επιλέξει τη βαθμολογία του (1 έως 5) σέρνοντας το δάχτυλό του ή κάνοντας κλικ στα αστέρια.
- Υποβολή: Το ViewModel στέλνει ένα HTTP POST αίτημα στο endpoint “/rating.php” του backend, μεταφέροντας το “parking_id” και το “rating”. Μετά την επιτυχή υποβολή, η οθόνη κλείνει αυτόματα με animation και ανανεώνεται η συνολική βαθμολογία του πάρκινγκ στην οθόνη λεπτομερειών.



Εικόνα 6.14 Οθόνη Αξιολόγησης (Light Theme)



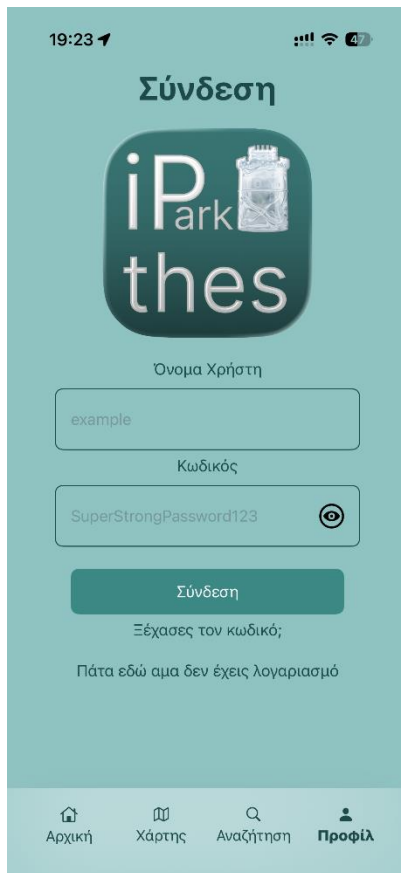
Εικόνα 6.15 Οθόνη Αξιολόγησης (Dark Theme)

6.3.7 Διαχείριση Λογαριασμού & Ασφάλεια

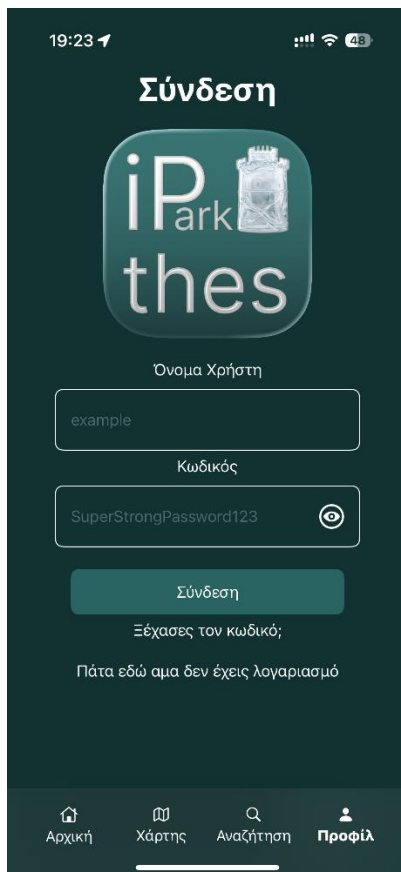
Αυτή η ενότητα περιλαμβάνει μια σειρά οθονών που διαχειρίζονται τον κύκλο ζωής του λογαριασμού χρήστη.

6.3.7.1 Διαχείριση Λογαριασμού & Ασφάλεια

- LoginView: Επιτρέπει την είσοδο στο σύστημα εισάγοντας όνομα χρήστη και κωδικό πρόσβασης. Περιλαμβάνει ελέγχους εγκυρότητας (π.χ. συμπλήρωση όλων των πεδίων) και επικοινωνεί με το “/user.php” στέλνοντας το Authentication Header (HTTP Basic Auth).

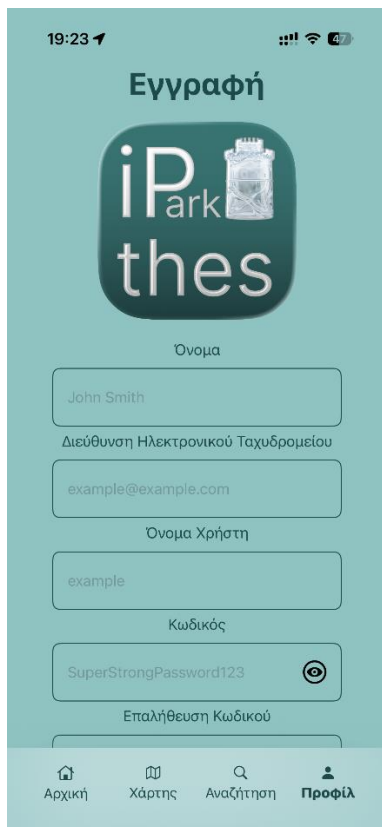


Εικόνα 6.16 Οθόνη Σύνδεσης (Light Theme)



Εικόνα 6.17 Οθόνη Σύνδεσης (Dark Theme)

- RegisterView: Επιτρέπει τη δημιουργία νέου λογαριασμού συλλέγοντας το ονοματεπώνυμο, τη διεύθυνση email, το όνομα χρήστη και τον κωδικό πρόσβασης. Πραγματοποιεί εκτενείς ελέγχους (ορθότητα μορφής email, ελάχιστο μήκος κωδικού).



Εικόνα 6.18 Οθόνη Εγγραφής (Light Theme)

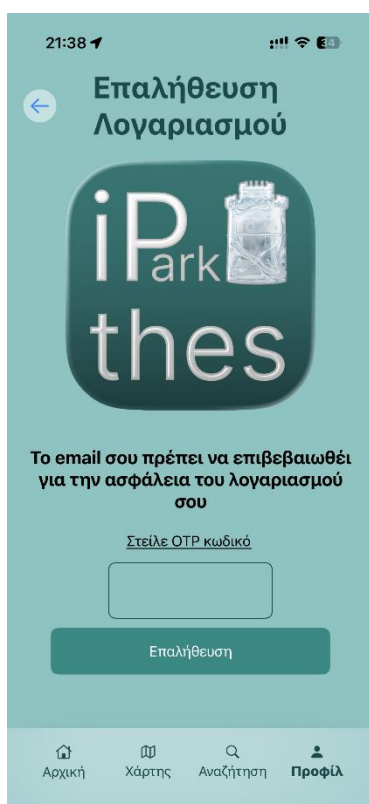


Εικόνα 6.19 Οθόνη Εγγραφής (Dark Theme)

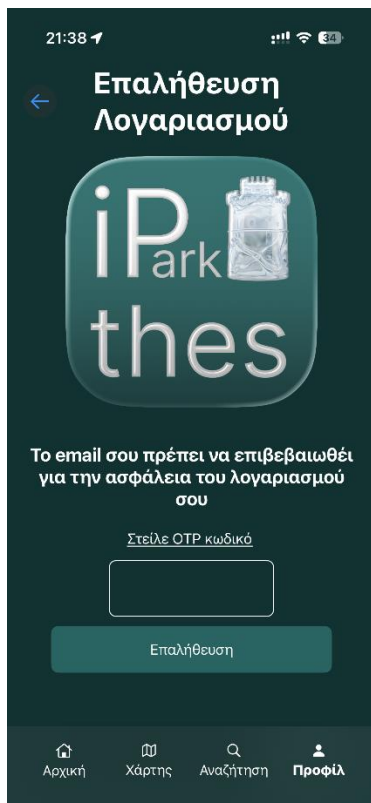
6.3.7.2 OTPView (One-Time Password Verification)

Μετά την εγγραφή ή κατά τη διάρκεια της ανάκτησης κωδικού, ο χρήστης μεταφέρεται στην οθόνη OTPView.

- **Λειτουργία:** Ο χρήστης καλείται να εισάγει έναν 6ψήφιο κωδικό που του στάλθηκε στο email. Η οθόνη διαθέτει 6 ξεχωριστά πεδία εισαγωγής κειμένου. Μόλις ο χρήστης πληκτρολογήσει ένα ψηφίο, η εστίαση (focus) μεταφέρεται αυτόματα στο επόμενο πεδίο (auto-focus shifting).
- **Ασφάλεια:** Η οθόνη περιλαμβάνει ένα χρονόμετρο αντίστροφης μέτρησης 10 λεπτών. Αν ο χρόνος παρέλθει, ο κωδικός λήγει και ο χρήστης πρέπει να ζητήσει νέο κωδικό μέσω του κουμπιού επαναποστολής. Η επαλήθευση γίνεται μέσω του “/verify.php” endpoint.



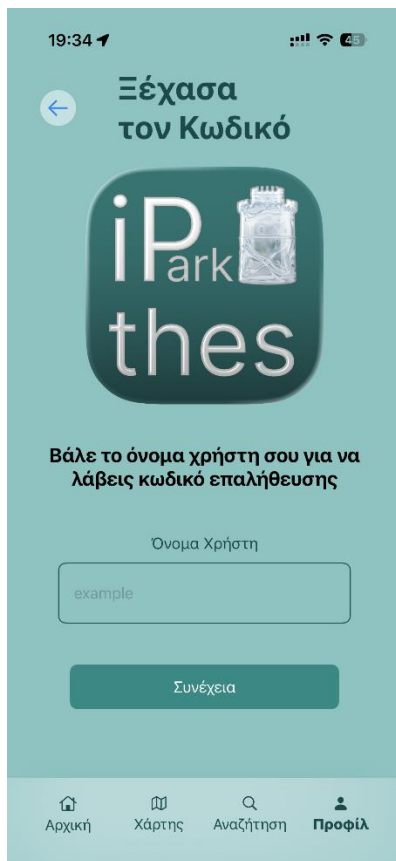
Εικόνα 6.20 Οθόνη OTP (Light Theme)



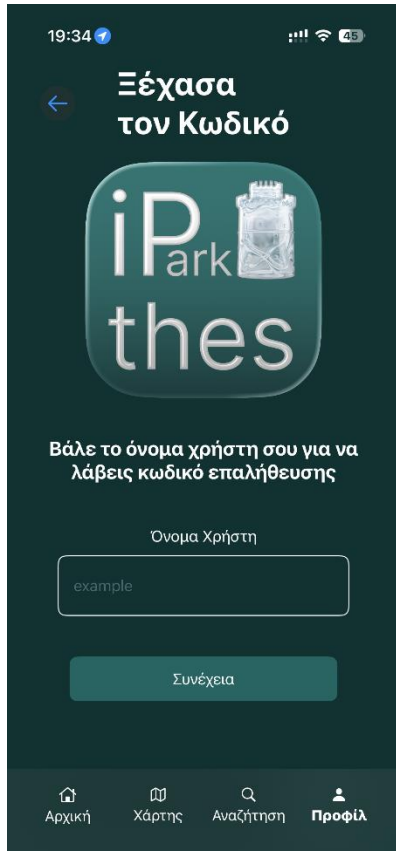
Εικόνα 6.21 Οθόνη OTP (Dark Theme)

6.3.7.3 ForgotPasswordView

Επιτρέπει την ανάκτηση του κωδικού πρόσβασης. Ο χρήστης εισάγει το όνομα χρήστη του, λαμβάνει έναν OTP κωδικό στο email του για να επιβεβαιώσει την ταυτότητά του, και στη συνέχεια εισάγει τον νέο κωδικό πρόσβασης. Η επικοινωνία γίνεται με το endpoint “/forgot_password.php”.



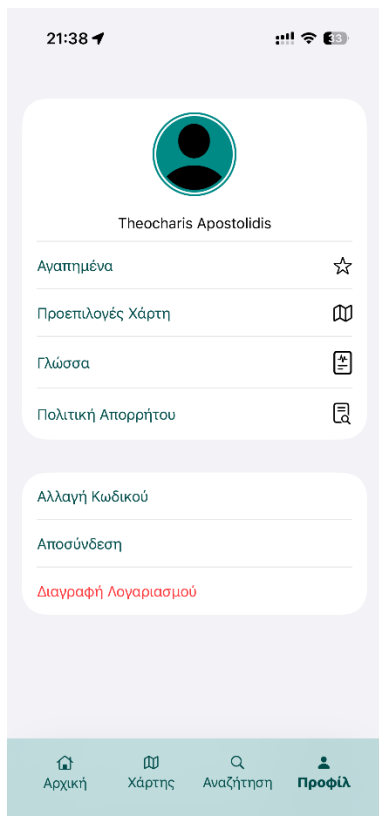
Εικόνα 6.22 Οθόνη Ανάκτησης Λογαριασμού (Light Theme)



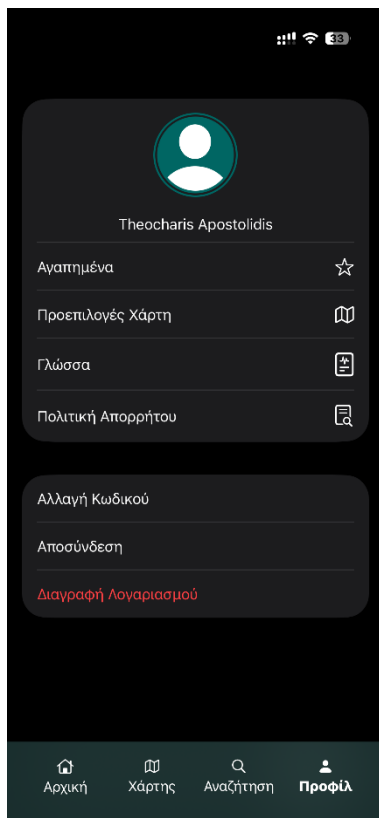
Εικόνα 6.23 Οθόνη Ανάκτησης Λογαριασμού (Dark Theme)

6.3.8 AccountView (Ρυθμίσεις)

Αυτή η οθόνη εμφανίζεται όταν ο χρήστης είναι συνδεδεμένος και επιλέξει την καρτέλα του λογαριασμού. Χρησιμοποιεί το native “List” style με “insetGrouped” εμφάνιση, ακολουθώντας τα πρότυπα των ρυθμίσεων του iOS (iOS Settings pattern).



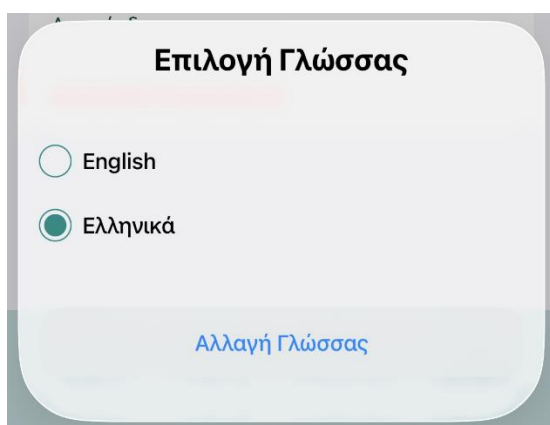
Εικόνα 6.24 Οθόνη Προφίλ (Light Theme)



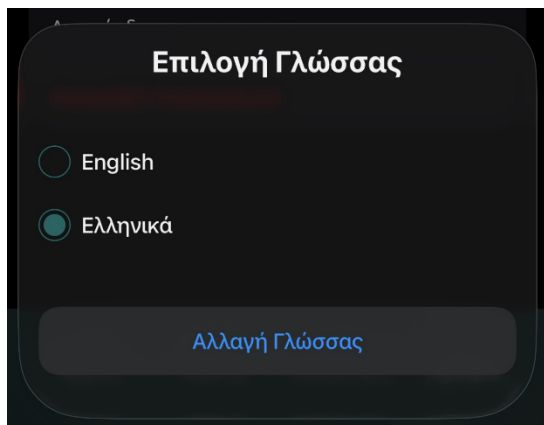
Εικόνα 6.25 Οθόνη Προφίλ (Dark Theme)

Λειτουργίες:

- Προβολή Προφίλ: Εμφανίζει το ονοματεπώνυμο του χρήστη μέσα σε στρογγυλεμένο πλαίσιο προφίλ.
- Αλλαγή Γλώσσας: Επιτρέπει τη μετάβαση σε οθόνη αλλαγής γλώσσας (“ChangeLanguageView”). Η εφαρμογή υποστηρίζει πλήρως Ελληνικά και Αγγλικά. Η αλλαγή εφαρμόζεται ακαριαία σε όλη την εφαρμογή χωρίς να απαιτείται επανεκκίνηση.

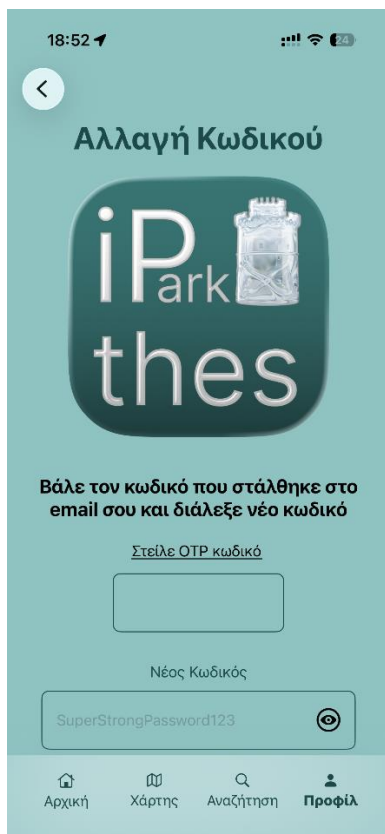


Εικόνα 6.26 Οθόνη Αλλαγής Γλώσσας (Light Theme)

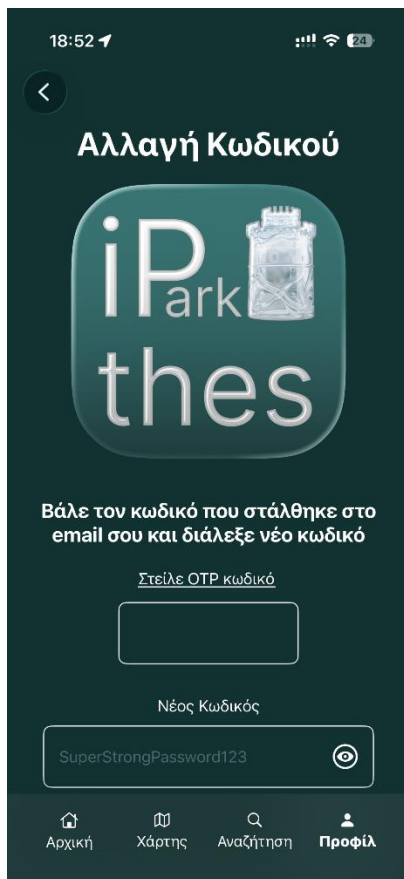


Εικόνα 6.27 Οθόνη Αλλαγής Γλώσσας (Dark Theme)

- Αλλαγή Κωδικού: Οδηγεί σε φόρμα αλλαγής κωδικού με αποστολή OTP για αυξημένη ασφάλεια.



Εικόνα 6.28 Οθόνη Αλλαγής Κωδικού (Light Theme)



Εικόνα 6.29 Οθόνη Αλλαγής Κωδικού (Dark Theme)

- Έξοδος (Logout): Αποσυνδέει τον χρήστη, διαγράφει τα διαπιστευτήρια από το Keychain και επαναφέρει το “loginStatus” σε “.guest”.
- Διαγραφή Λογαριασμού (Delete Account): Μια εξαιρετικά κρίσιμη λειτουργία ασφαλείας. Εμφανίζει μια κόκκινη προειδοποιητική ειδοποίηση (destructive alert). Εάν ο χρήστης επιβεβαιώσει τη διαγραφή, η εφαρμογή στέλνει αίτημα διαγραφής στο backend (το οποίο λόγω του “ON DELETE CASCADE” στη βάση δεδομένων διαγράφει αυτόματα όλα τα αγαπημένα και τις αξιολογήσεις του χρήστη) και διαγράφει οριστικά τα δεδομένα του από τη συσκευή.

6.4 Σύστημα Εντοπισμού Τοποθεσίας (CoreLocation & LocationManager)

Ένα από τα πιο σημαντικά υποσυστήματα της εφαρμογής είναι η κλάση “LocationManager”, η οποία αποτελεί έναν wrapper γύρω από το native framework “CoreLocation” της Apple.

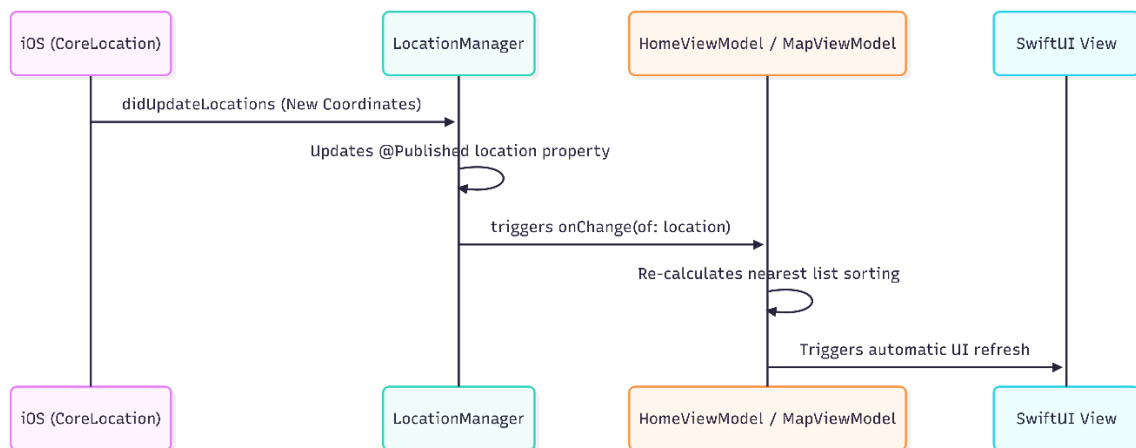
6.4.1 Υλοποίηση του LocationManager

Η κλάση έχει σχεδιαστεί ως ένα “ObservableObject” και εκτίθεται σε όλη την εφαρμογή ως “EnvironmentObject” προκειμένου να παρέχει άμεση πρόσβαση στις γεωγραφικές συντεταγμένες του χρήστη χωρίς περιττά instantiation.

Τα βασικά καθήκοντα του “LocationManager” είναι:

- Αίτημα Δικαιωμάτων: Ζητά άδεια πρόσβασης στην τοποθεσία του χρήστη κατά την πρώτη εκκίνηση ("When In Use" authorization).
- Συνεχής Ενημέρωση (Location Tracking): Λαμβάνει τις συντεταγμένες και ενημερώνει την “Published” τιμή location.
- Υπολογισμός Απόστασης: Παρέχει helper μεθόδους για τον υπολογισμό της απόστασης σε μέτρα ή χιλιόμετρα μεταξύ του χρήστη και ενός πάρκινγκ, κάτι που χρησιμοποιείται για την ταξινόμηση στην αρχική οθόνη και στη λίστα αναζήτησης.

Η ροή ενημέρωσης τοποθεσίας λειτουργεί αντιδραστικά (reactively):



Εικόνα 6.30 Sequence Diagram της ενημέρωσης τοποθεσίας

6.5 Ασφάλεια Δεδομένων στη Συσκευή (iOS Keychain)

Για την αποθήκευση ευαίσθητων δεδομένων του χρήστη, όπως το όνομα χρήστη (username) και ο κρυπτογραφημένος κωδικός πρόσβασης (password token) που χρησιμοποιούνται για το HTTP Basic Authentication, η εφαρμογή iParkThess αποφεύγει τη χρήση του “UserDefaults” (το οποίο αποθηκεύει δεδομένα σε απλή μορφή XML στο δίσκο).

Αντίθετα, χρησιμοποιεί την κλάση Keychain, η οποία αλληλοεπιδρά με το ασφαλές σύστημα αποθήκευσης της συσκευής (iOS Keychain). Το Keychain χρησιμοποιεί κρυπτογράφηση επιπέδου hardware (Secure Enclave) και διασφαλίζει ότι τα αποθηκευμένα διαπιστευτήρια δεν μπορούν να διαβαστούν από άλλες εφαρμογές ή σε περίπτωση παραβίασης της συσκευής (jailbreak). Αυτό εγγυάται ότι η αυτόματη σύνδεση (auto-login) πραγματοποιείται με απόλυτη ασφάλεια [13].

Κεφάλαιο 7ο: Συμπεράσματα / Προτάσεις Βελτίωσης

7.1 Συμπεράσματα

Το έργο που αναλύθηκε παραπάνω αποδείχτηκε πιο απαιτητικό από ότι περίμενα μιας και χρειάστηκε να επενδυθεί χρόνος για την κατανόηση των τεχνολογιών που χρησιμοποιήθηκαν και συγκέντρωση για την απασφαλμάτωση των προβλημάτων που προέκυψαν κατά την ανάπτυξη. Όμως καταλήγω στο συμπέρασμα ότι άξιζε η ανάπτυξη μιας εφαρμογής για την εύρεση θέσεων στάθμευσης στην πόλη της Θεσσαλονίκης γιατί είναι ένα κενό στην αγορά, και έχει νόημα. Είναι μια εφαρμογή που σίγουρα θα παραμείνει εγκατεστημένη και θα χρησιμοποιηθεί από αρκετά άτομα στον περίγυρο μου, σε περίπτωση που γίνει διαθέσιμη στο κοινό. Τέλος, για την βεβαίωση τήρησης ορθού ακαδημαϊκού λόγου, χρησιμοποιήθηκε η βοήθεια τεχνητής νοημοσύνης.

7.2 Προτάσεις Βελτίωσης

Παρ' όλο που το έργο είναι ολοκληρωμένο από την μεριά του client / backend όπου και είναι το scope της Π. Ε., δεν σημαίνει ότι δεν έχει περιθώρια περεταίρω ανάπτυξης, για παράδειγμα το κομμάτι των επιχειρήσεων. Οι επιχειρήσεις που προσφέρονται και διαφημίζονται στην εφαρμογή iParkThess δεν έχουν κάποιο τρόπο να ενημερώνουν τα στοιχεία τους σε πραγματικό χρόνο. Πρέπει να ενημερώνεται η βάση χειροκίνητα.

Με την ανάπτυξη ενός πληροφοριακού συστήματος (Content Management System) θα μπορέσει η κάθε επιχείρηση να ενημερώνει μόνη της, είτε αυτόματα είτε χειροκίνητα την διαθεσιμότητά της καθώς και τις παροχές της. Σαφώς ένα τέτοιο σύστημα καθιστά ένα έργο από μόνο του λόγω της πολυπλοκότητάς του τόσο στην έρευνα της αγοράς όσο και στην υλοποίηση και αφομοίωση του χωρίς πολλές αλλαγές από τις επιχειρήσεις.

ΒΙΒΛΙΟΓΡΑΦΙΑ

- [1] FAQ: FuelGR. <https://fuelgr.gr/faq>
- [2] PHP.Watch: PHP JIT in Depth <https://php.watch/articles/jit-in-depth>
- [3] PHP: MySQL Improved Extension. <https://www.php.net/manual/en/book.mysqli.php>
- [4] PHP Manual: <https://www.php.net/manual/en/index.php>
- [5] InnoDB vs MyISAM: A detailed comparison of two MySQL storage engines:
<https://www.managedserver.eu/innodb-vs-myisam-a-detailed-comparison-of-two-mysql-storage-engines/>
- [6] Protobuf Manual: <https://protobuf.dev/>
- [7] Amato, Nicola. Mastering database normalization: A comprehensive exploration of normal forms. October 2023
- [8] The Swift Programming language: <https://docs.swift.org/swift-book/documentation/the-swift-programming-language/>
- [9] Ranjith Kumar Vanaparthi. From UIKit to SwiftUI: A Quantitative Analysis of Migration Strategies for Large-Scale Financial Application. February 2026
- [10] SwiftUI Documentation: <https://developer.apple.com/documentation/swiftui>
- [11] UIKit Documentation <https://developer.apple.com/documentation/uikit>
- [12] Building an iOS App with MVVM and Clean Architecture: A Practical Guide: <https://medium.com/@krishnaprakashn/building-an-ios-app-with-mvvm-and-clean-architecture-a-practical-guide-16c0fd74ca44>
- [13] Apple Platform Security: Keychain Data Protection. <https://support.apple.com/guide/security/keychain-data-protection-secb0694df1a>