ΔΙΕΘΝΕΣ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΤΗΣ ΕΛΛΑΔΟΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

«Εφαρμογή εύρεσης ιδιωτικών σταθμών στάθμευσης»

Του φοιτητή
Σμπήλια Βασίλειου
Αρ. Μητρώου: 164744

Επιβλέπων
Ιγνάτιος Δεληγιάννης
Καθηγητής

Ημερομηνία 30/08/2022

Τίτλος Δ.Ε.Εφαρμογή εύρεσης ιδιωτικών σταθμών στάθμευσης
Κωδικός Δ.Ε. 21175
Ονοματεπώνυμο φοιτητή Βασίλειος Σμπήλιας
Ονοματεπώνυμο εισηγητή Ιγνάτιος Δεληγιάννης
Ημερομηνία ανάληψης Δ.Ε. 09-03-2021
Ημερομηνία περάτωσης Δ.Ε. 30/08/2022

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 διπλωματική εργασία, στο Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του ΔΙ.ΠΑ.Ε.

«Σε όσους με στήριξαν»

Πρόλογος

Όντας φοιτητής στην Θεσσαλονίκη από τον πρώτο καιρό συνειδητοποίησα, ότι η πόλη έχει σοβαρό πρόβλημα με τους χώρους στάθμευσης και αυτό αποτέλεσε πηγή έμπνευσης για την σύλληψη της ιδέας της δημιουργίας μιας εφαρμογής για την εύρεση ιδιωτικών σταθμών στάθμευσης οχημάτων.

Πλέον οι άνθρωποι αναζητούν πληροφορίες χρησιμοποιώντας τους υπολογιστές και τα κινητά τους τηλέφωνα. Έτσι είναι πολύ απλό, εφόσον για παράδειγμά κάποιος επιθυμεί να πάει στο κέντρο της πόλης, αντί να χάσει πολύτιμο χρόνο για την εύρεση θέσης στάθμευσης του οχήματος του, να αναζητήσει μέσω της εφαρμογής έναν σταθμό στάθμευσης, να πλοηγηθεί μέχρι αυτόν και αν το επιθυμεί να κάνει και κράτηση θέσης στον σταθμό αυτόν.

Περίληψη

Ο στόχος της συγκεκριμένης διπλωματικής εργασίας είναι η ανάγκη εύρεσης ιδιωτικών σταθμών στάθμευσης, που θα δίνει την δυνατότητα στον χρήστη της εφαρμογής να δημιουργεί λογαριασμό και να μπορεί να αναζητήσει στον χάρτη ιδιωτικούς σταθμούς στάθμευσης.

Ο χρήστης αφού επιλέξει στον χάρτη κάποιον σταθμό εμφανίζονται κάποιες πληροφορίες σχετικά με αυτόν όπως η διεύθυνσή του, το τηλέφωνο επικοινωνίας, το κόστος στάθμευσης ανά ώρα του σταθμού καθώς και οι διαθέσιμες θέσεις στάθμευσης που έχει, καθώς και τρεις επιλογές όπως η αναζήτηση της συντομότερης διαδρομής από την τρέχουσα τοποθεσία έως τον επιλεγμένο σταθμό και την πλοήγηση του, την αποθήκευση του σταθμού στα αγαπημένα του χρήστη και την επιλογή της κράτησης θέσης.

Εκτός από το μέρος της εφαρμογής που αναφέρεται στον χρήστη υπάρχει και αυτό του διαχειριστή όπου ο ιδιοκτήτης ενός σταθμού στάθμευσης μπορεί να διαχειρίζεται και να επεξεργάζεται τις πληροφορίες του σταθμού, όπως και τις διαθέσιμες θέσεις του.

«Application for finding private parking»

«Vasileios Smpilias»

Abstract

The aim of this thesis is the need to find private parking stations that will enable the user of this application to create an account and be able to search for private parking stations on the map.

After the user selects a station on the map, some information about the specific station is displayed, such as its address, contact phone number, parking coast per hour of the station, as well as the available parking seats for the station, as well as three options such as searching for the shortest route from the current location of the user to the selected station and its navigation, saving the station to the user's favorites and selecting to booking a seat of this station.

In addition to the user part in the application there is also the administrator part where the owner of the parking station can manage and edit the information of the station as well as its available parking seats.

Ευχαριστίες

Θα ήθελα να ευχαριστήσω θερμά την οικογένεια μου και κυρίως τους γονείς μου που με στήριξαν ψυχολογικά και οικονομικά σε όλη την διάρκεια των σπουδών μου. Καθώς και τον κ. Δεληγιάννη που με καθοδήγησε και ήταν διαθέσιμος όποτε χρειαζόταν.

Περιεχόμενα

Κατάλογος Εικόνων

Κεφάλαιο 1ο: Τεχνολογίες που χρησιμοποιήθηκαν

1.1 Εισαγωγή

Για την εκπόνηση της παρούσας εργασίας χρησιμοποιήθηκαν βασικές τεχνολογίες και εργαλεία του Android Studio και της Google. Η ανάπτυξη της εφαρμογής έγινε στο Android Studio με χρήση της γλώσσας Java. Για την σύνδεση του χρήση στην εφαρμογή, την δημιουργία λογαριασμού και την αποθήκευση δεδομένων του χρήστη αλλά και την αποθήκευση των κρατήσεων καθώς και την αποθήκευση των πληροφοριών των σταθμών χρησιμοποιήθηκε η βάση δεδομένων Firebase της Google. Για την εμφάνιση των σταθμών και της τρέχουσας τοποθεσίας του χρήστη χρησιμοποιήθηκε το google maps και συγκεκριμένα το API Maps SDK for Android σε συνδυασμό με το Direction API για την εύρεση της συντομότερης διαδρομής μεταξύ της τρέχουσας τοποθεσίας του χρήστη και του σταθμού της επιλογής του.

1.2 Λογισμικό Android.

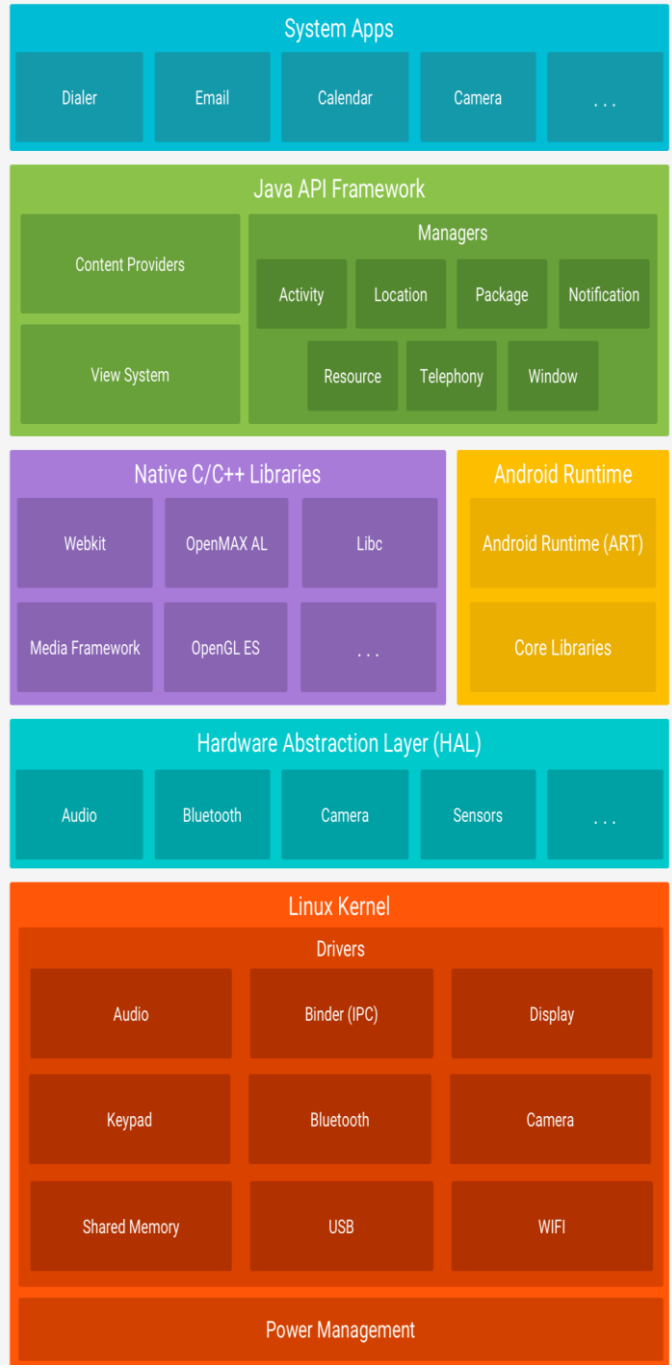
Το Android είναι ένα open source λειτουργικό σύστημα σχεδιασμένο κατά κύριο λόγο για συσκευές με οθόνη αφής, όπως, συσκευές κινητής τηλεφωνίας και tablets, βασισμένο σε πυρήνα Linux. Η ανάπτυξη του γίνεται από την Google και πρωτοπαρουσιάστηκε το 2007 από την Open Handset Alliance.

Μερικά από τα χαρακτηριστικά του Android είναι τα εξής:

- Οθόνη αφής πολλαπλών σημείων.
- Υποστήριξη Java.
- Υποστήριξη πολυμέσων.
- Περιήγηση στο διαδίκτυο.
- Αγορά και εγκατάσταση εφαρμογών.
- Συνδεσιμότητα.
- Αποστολή μηνυμάτων.

1.3 Αρχιτεκτονική Android

Το Android είναι μια στοίβα λογισμικού open source βασισμένη σε Linux. Η αρχιτεκτονική του android αποτελείται από τα παρακάτω έξι επίπεδα, όπως φαίνονται στην Εικόνα 1.



Εικόνα 1: Αρχιτεκτονική Android

1.3.1 Πυρήνας Linux

Αρχίζοντας από κάτω προς τα επάνω στη βάση αρχιτεκτονικής του Android συναντάμε τον πυρήνα Linux. Ο πυρήνας είναι υπεύθυνος για την επικοινωνία μεταξύ Hardware και Software. Χωρίς την ύπαρξη του πυρήνα δεν υπάρχει η δυνατότητα για ένα αξιόπιστο λειτουργικό σύστημα. Ο πυρήνας είναι υπεύθυνος για την εκτέλεση των βασικών λειτουργιών του συστήματος όπως η διαχείριση της μνήμης, η διαχείριση των διεργασιών, η χρήση υπηρεσιών δικτύωσης και η χρήση οδηγών για διάφορα τμήματα του Hardware.

1.3.2 Hardware Abstraction Layer (HAL)

Το HAL παρέχει interfaces που υποστηρίζουν τις δυνατότητες των εξαρτημάτων μιας συσκευής, όπως, κάμερα, ηχεία, μικρόφωνα κ.α. Το HAL αποτελείται από πολλές βιβλιοθήκες όπου η κάθε μία υλοποιεί ένα interface για έναν συγκεκριμένο τύπο υλικού. Το σύστημα Android φορτώνει την λειτουργική μονάδα της κατάλληλης βιβλιοθήκης για κάποιο υλικό όταν η εφαρμογή κάνει κλήση για πρόσβαση στο συγκεκριμένο υλικό της συσκευής.

1.3.3 Βιβλιοθήκες

Πάνω από τον πυρήνα του Linux βρίσκεται ένα σύνολο από βιβλιοθήκες γραμμένες σε C και C++ οι οποίες χρησιμοποιούνται από διάφορα στοιχεία του συστήματος του Android από τα υψηλότερα επίπεδα.

1.3.4 Android Runtime

Για τις συσκευές με έκδοση Android 5.0 και πάνω η κάθε μία εφαρμογή εκτελείται με την δική της διαδικασία και με το δικό της στιγμιότυπο στο Android Runtime. Το Android Runtime έχει την δυνατότητα να μεταφράζει την εφαρμογή και να αποθηκεύει την μετάφραση ώστε αυτή να είναι πάντα διαθέσιμη για εκτέλεση στο λειτουργικό σύστημα. Αυτό έχει ως πλεονέκτημα ο απαιτούμενος χρόνος για να ανοίξουν οι εφαρμογές να είναι μικρότερος, να γίνεται καλύτερη διαχείριση της μνήμης της συσκευής και να καταναλώνεται λιγότερη ενέργεια.

1.3.5 Java API Framework

Στο επίπεδο πλαισίου των εφαρμογών οι προγραμματιστές έχουν την δυνατότητα να αναπτύξουν εφαρμογές με την χρήση του Android. Το επίπεδο αυτό είναι ένα σύνολο υπηρεσιών που αποτελούν συλλογικά το περιβάλλον όπου οι εφαρμογές τρέχουν.

Το Android περιλαμβάνει τις ακόλουθες βασικές υπηρεσίες:

- **Content Providers.** Επιτρέπει στις εφαρμογές να ανταλλάσσουν δεδομένα με άλλες εφαρμογές.
- **View System.** Το View System είναι ένα σύνολο από γραφικά στοιχεία για τη δημιουργία των διεπαφών χρήστη.
- **Activity Manager**. Ελέγχει τον κύκλο ζωής των εφαρμογών και παρέχει τη δυνατότητα μετάβασης σε προγενέστερες καταστάσεις τους.
- **Location Manager**. Παρέχει πρόσβαση στις υπηρεσίες εντοπισμού θέσης μέσω GPS και επιτρέπει μια εφαρμογή να λαμβάνει ενημερώσεις σχετικά με τις αλλαγές της τοποθεσίας.
- **Package Manager**. Επιτρέπει στις εφαρμογές να μάθουν πληροφορίες σχετικά με άλλες εφαρμογές που είναι εγκατεστημένες στη συσκευή.
- **Notifications Manager**. Επιτρέπει στις εφαρμογές να εμφανίζουν ειδοποιήσεις στον χρήστη.
- **Resource Manager**. Παρέχει πρόσβαση σε ενσωματωμένους πόρους όπως ρυθμίσεις χρωμάτων και layouts.
- **Telephony Manager**. Παρέχει πληροφορίες στην εφαρμογή σχετικά με τις υπηρεσίες τηλεφωνίας και κλήσεων που υπάρχουν στη συσκευή.
- **Window Manager**. Διαχειρίζεται τα ανοιχτά παράθυρα ώστε κάθε activity να εμφανίζεται σε ένα παράθυρο.

1.3.6 Επίπεδο Εφαρμογής

Στο υψηλότερο επίπεδο της αρχιτεκτονικής του Android βρίσκεται το επίπεδο εφαρμογής όπου υπάρχουν οι εφαρμογές που χρησιμοποιεί ένας χρήστης. Οι εφαρμογές αυτές μπορεί να είναι οι βασικές εφαρμογές όπως κλήσεις, χάρτες, κάμερα, συλλογή, ημερολόγιο, αριθμομηχανή καθώς και εφαρμογές που επιλέγονται από τον χρήστη μέσω του Google Play Store.

1.4 Android Studio

Το Android Studio είναι το επίσημο ολοκληρωμένο περιβάλλον ανάπτυξης (IDE) της Google για την ανάπτυξη εφαρμογών Android βασισμένο στο IntelliJ IDEA. Εκτός από τα εργαλεία του IntelliJ το Android Studio προσφέρει και άλλες λειτουργίες που βελτιώνουν την ανάπτυξη Android εφαρμογών όπως:

- Ευέλικτο σύστημα κατασκευής με βάση το Gradle.
- Εξομοιωτή γρήγορο και πλούσιο σε χαρακτηριστικά.
- Ενιαίο περιβάλλον όπου μπορεί να γίνει ανάπτυξη για όλες τις συσκευές Android.
- Πρότυπα κώδικα και χρήση του Github για την δημιουργία βιβλιοθηκών και την εισαγωγή sample code.
- Ειδικά εργαλεία για debug και test.
- Υποστήριξη C++.
- Ενσωματωμένη υποστήριξη για την πλατφόρμα Google Cloud.

1.5 Βασικά Συστατικά Εφαρμογών Android

Μια Android εφαρμογή αποτελείται από τα παρακάτω τέσσερα βασικά δομικά στοιχεία.

- Activities
- Services
- Broadcast receivers
- Content providers

Κάθε ένα από τα παραπάνω εξυπηρετεί έναν σκοπό και έχει ξεχωριστό κύκλο ζωής.

1.5.1 Activities

Το Activity είναι το πιο διαδεδομένο δομικό στοιχείο που συναντάται σε μια εφαρμογή Android καθώς αναπαριστά και μια ξεχωριστεί ‘οθόνη’ μιας εφαρμογής. Όλες οι εφαρμογές Android απαρτίζονται από μια συλλογή από activities. Τα activities αυτά οργανώνονται σε μια στοίβα και στην πρώτη θέση της στοίβας βρίσκεται η οθόνη που βλέπει εκείνη την στιγμή ο χρήστης.

Όταν μια εφαρμογή ξεκινάει στην οθόνη του χρήστη εμφανίζεται το Main Activity εκτός αν έχουμε κάποια εντολή στο Main Activity να εμφανίζεται κάποιο άλλο Activity. Όταν ένας χρήστης μεταφέρεται σε κάποιο άλλο Activity αυτό μεταφέρεται στην κορυφή της στοίβας. Αυτό συμβαίνει κάθε φορά που ο χρήστης μεταφέρεται σε νέο Activity.

Τα Activities περιλαμβάνουν views, τα οποία βοηθούν τον χρήστη να αλληλεπιδρά με το κάθε Activity και την εφαρμογή. Μερικά από τα views είναι τα εξής:

- TextView
- Button
- ImageButton
- ScrollView

- GridView
- RecyclerView
- FragmentCointainerView

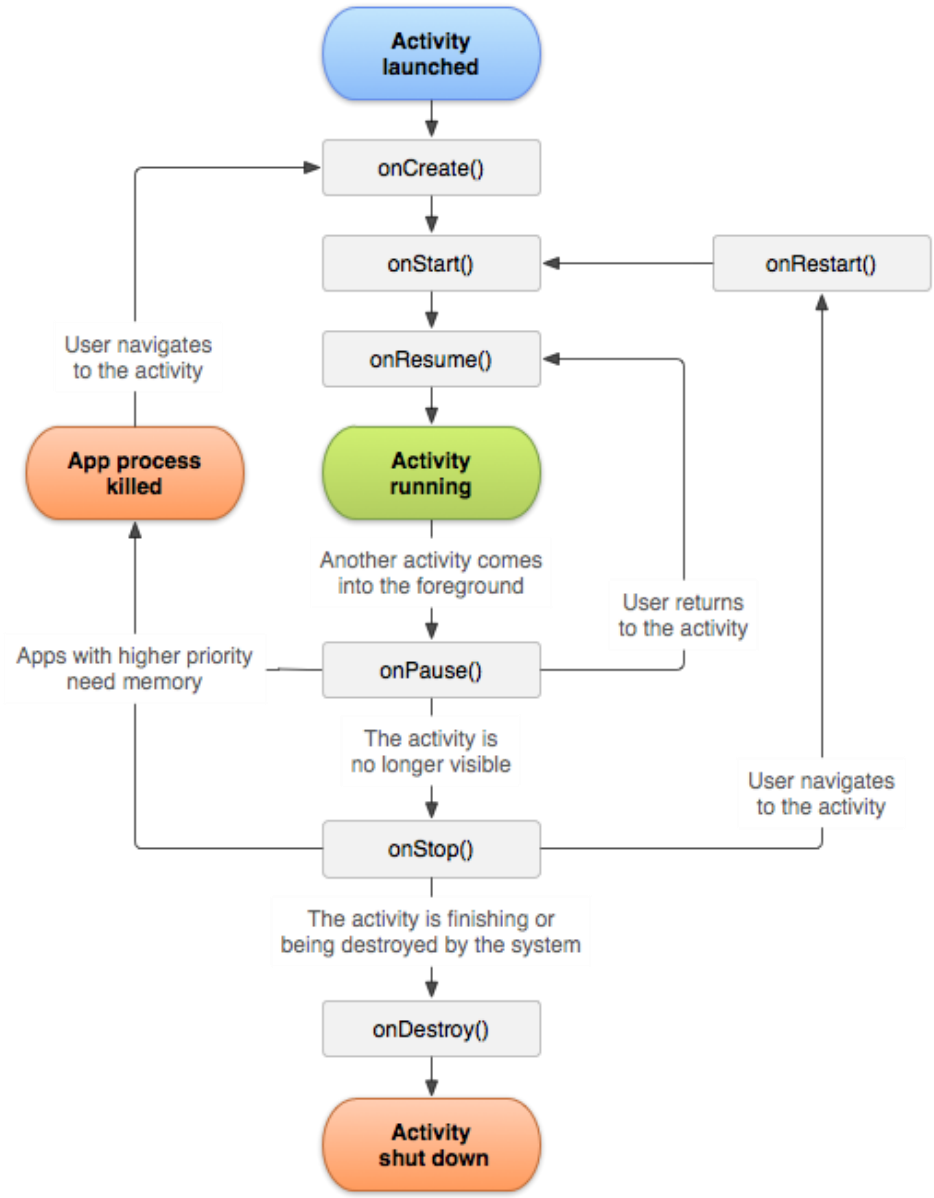
Κάθε Activity ενεργοποιείται με την βοήθεια ενός Intent. Κάθε Intent είναι μια αφηρημένη περιγραφή μιας πράξης που πρόκειται να εκτελεστεί και χρησιμοποιείται για την εκκίνηση ενός Activity ή και για να περάσουμε δεδομένα από ένα Activity σε ένα νέο.

1.5.1.1 Ο κύκλος ζωής ενός Activity

Ο κύκλος ζωής ενός Activity από τρεις βασικές καταστάσεις οι οποίες είναι οι εξής:

- **Activity ενεργό**. Σε αυτήν την κατάσταση το Activity είναι ενεργό και ορατό προς τον χρήστη.
- **Activity σε παύση**. Όταν ένα Activity βρίσκεται σε παύση έχει σταματήσει προσωρινά τη λειτουργία του γιατί ένα άλλο Activity βρίσκεται στο προσκήνιο, δηλαδή στην κορυφή της στοίβας.
- **Activity σταματημένο**. Όταν ένα Activity είναι σταματημένο συνεχίζει να διατηρεί τις πληροφορίες που είχε και βρίσκεται στην κατάσταση που ήταν πριν σταματήσει αλλά σε περίπτωση που κάποια άλλη εφαρμογή χρειαστεί τη μνήμη που καταλαμβάνει αυτό το Activity τότε το Activity πεθαίνει.

Στην Εικόνα 2 φαίνεται ο κύκλος ζωής ενός Activity. Ο κύκλος ζωής του Activity ξεκινάει με την μέθοδο onCreate() όπου το Activity έρχεται σε ζωή και η δραστηριότητα του τελειώνει με την μέθοδο onDestroy() όπου το Activity καταστρέφεται (πεθαίνει). Ο χρόνος που ένα Activity είναι ορατό προς τον χρήστη ξεκινάει με την μέθοδο onStart() και σταματάει με την μέθοδο onStop(). Η αλληλεπίδραση του χρήστη με ένα Activity γίνεται όταν αυτό βρίσκεται ανάμεσα στις μεθόδους onResume() και onPause(). Η μέθοδος onPause() καλείται όταν μια συσκευή βρίσκεται σε αδράνεια ή όταν ξεκινάει μια άλλη δραστηριότητα, ενώ κατά την επιστροφή στην προηγούμενη δραστηριότητα καλείται η μέθοδος onResume().



Εικόνα 2: Ο κύκλος ζωής ενός Activity

1.5.2 Services

Η υπηρεσία(Service) εκκινείται από κάποιο άλλο δομικό στοιχείο. Μια υπηρεσία τρέχει background, ενώ ο χρήστης μπορεί να το γνωρίζει και ανά πάσα στιγμή να την σταματήσει, ενώ μπορεί ο χρήστης να μην το γνωρίζει και το σύστημα να είναι αυτό το οποίο μπορεί να την διαχειριστεί. Δηλαδή το σύστημα εάν υπάρχει ανάγκη μνήμης μπορεί να την σταματήσει και να την εκκινήσει ξανά αργότερα.

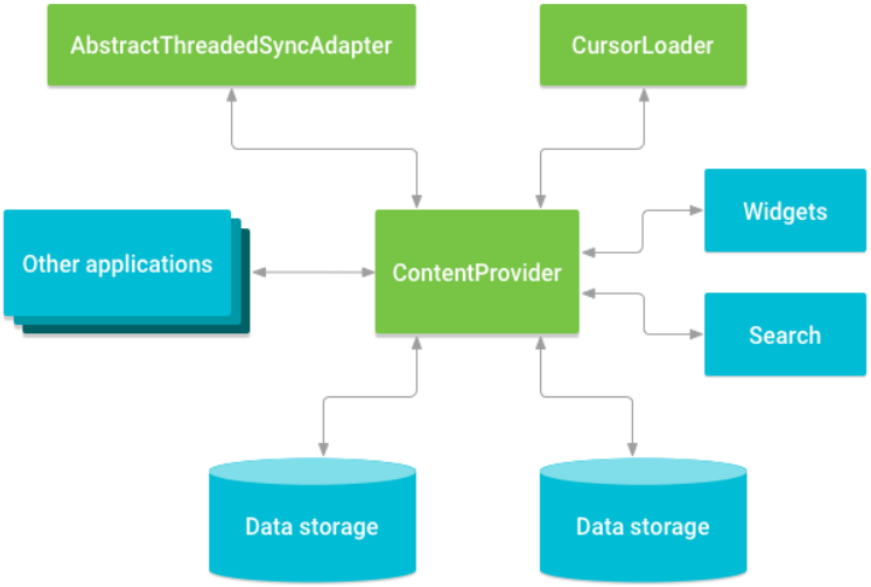
1.5.3 Broadcast receivers

Τα Broadcast receivers ενεργοποιούνται από το λειτουργικό σύστημα όταν συμβαίνει κάποιο γεγονός όπως η ολοκλήρωση λήψης ενός αρχείου μιας εφαρμογής ή η ένδειξη χαμηλής μπαταρίας καθώς και η λήψη ενός μηνύματος ή μιας αναπάντητης κλήσης. Ένα Broadcast receiver μπορεί να ξυπνήσει ένα activity ή και να βγάλει στον χρήστη μια ειδοποίηση.

1.5.4 Content Provider

Οι Content Provider διαχειρίζονται τα δεδομένα και τα αρχεία που αποθηκεύονται στο σύστημα αρχείων της συσκευής. Αυτό το σύστημα αρχείων μπορεί να είναι είτε μια βάση δεδομένων SQLite είτε η Firebase της Google που χρησιμοποιείται στην παρούσα Δ.Ε. είτε οποιοδήποτε μέσο μόνιμης αποθήκευσης.

Στην Εικόνα 3 φαίνεται η σχέση του Content Provider και άλλων components.



Εικόνα 3: Η σχέση του Content Provider μεταξύ άλλων components

1.6 Firebase

Για το authentication του χρήστη και για την αποθήκευση των δεδομένων όπως των σταθμών στάθμευσης των κρατήσεων και των στοιχείων του χρήστη χρησιμοποιήθηκε η διαδικτυακή βάση δεδομένων Firebase της Google. Η Firebase είναι εύχρηστη και συνεργάζεται με τις εφαρμογές Android. Για την εφαρμογή χρησιμοποιήθηκε η Firebase Firestore και η Realtime database οι οποίες είναι NoSQL βάσεις δεδομένων που σημαίνει ότι τα δεδομένα αποθηκεύονται σε αρχεία που περιέχουν πεδία αντιστοιχούμενα με τιμές.

Τα κύρια χαρακτηριστικά της Firebase είναι τα εξής:

- **Αυθεντικοποίηση του χρηστη**: Ένας χρήστης μπορεί να συνδεθεί με email και password είτε με Google, Facebook, GitHub κ.α.
- **Ευελιξία**: Η Firebase υποστηρίζει ευέλικτες δομές δεδομένων. Τα δεδομένα μπορούν να αποθηκεύονται σε αρχεία οργανωμένα σε συλλογές.
- **Realtime ενημερώσεις**: Τα δεδομένα ενημερώνονται σε όλες τις συσκευές σε πραγματικό χρόνο.
- **Υποστήριξη εκτός δικτύου**: Στο Firestore τα δεδομένα αποθηκεύονται σε οποιαδήποτε συσκευή ακόμα και αν δεν υπάρχει σύνδεση στο διαδίκτυο. Ο συγχρονισμός όλων των αλλαγών μπορεί να γίνει όταν επανέρθει η σύνδεση στο δίκτυο.

1.7 Java

Η Java είναι μία αντικειμενοστρεφής γλώσσα προγραμματισμού η οποία σχεδιάστηκε από την εταιρεία Sun Microsystems που αργότερα εξαγοράστηκε από την Oracle. Η Java χρησιμοποιείται κυρίως για Desktop εφαρμογές, back-end προγραμματισμό και για Android εφαρμογές.

1.7.1 Βασικά Χαρακτηριστικά της γλώσσας Java

Τα βασικά χαρακτηριστικά της γλώσσας προγραμματισμού Java είναι τα εξής:

- Είναι σχεδιασμένη για αντικειμενοστρεφή προγραμματισμό.
- Ο κώδικας της τρέχει παντού, σε όλες τις πλατφόρμες.
- Διαχειρίζεται αυτόματα τη μνήμη
- Είναι γλώσσα υψηλού επιπέδου
- Υποστηρίζει πολυνηματική επεξεργασία (multithreaded processing)
- Είναι δυναμική, δηλαδή προσαρμόζεται εύκολα σε διαφορετικά περιβάλλοντα και απαιτήσεις.

1.8 Geolocation στο Android

Η εύρεση της τοποθεσίας μιας συσκευής στο Android απαιτεί την κατασκευή ενός αντικειμένου της κλάσης LocationManager και ενός αντικειμένου της κλάσης LocationListener. Με το LocationListener υλοποιείται η διεπαφή LocationListener καθώς και οι παρακάτω μέθοδοι που χρησιμοποιούνται για την διαχείριση της τοποθεσίας του χρήστη.

- **onLocationChanged**(). Περιλαμβάνει ένα object της κλάσης Location που περιέχει την θέση του χρήστη.
- **onStatusChanged()**. Περιλαμβάνει πληροφορίες σχετικά με την κατάσταση σύνδεσης του πάρουχου ανάκτησης θέσης.
- **onProviderEnabled()**. Καλείται όταν ο πάροχος ανάκτησης θέσης ενεργοποιείται από τον χρήστη.
- **onProviderDisabled()**. Καλείται όταν ο πάροχος ανάκτησης θέσης απενεργοποιείται από τον χρήστη.

1.9 Google Maps

Οι χάρτες της Google πρωτοεμφανίστηκαν τον Φεβρουάριο του 2005 και είναι μια υπηρεσία χαρτογράφησης. Παρέχει χάρτες δρόμων και χάραξη διαδρομών για διάφορα είδη μεταφορών όπως με τα πόδια, με αυτοκίνητο κ.α. Επίσης οι χάρτες Google περιλαμβάνουν αναζήτηση σημείων, πόλεων, διευθύνσεων και παρέχουν και λειτουργία street view. Οι χάρτες της Google είναι διαθέσιμες στους προγραμματιστές μέσω των υπηρεσιών Google Maps API(Application Programming Interface).

Για πρόσβαση στο Google Maps API πρέπει να γίνει ενεργοποίηση λογαριασμού στο Google Cloud. Τα διαθέσιμα API κοστολογούνται ανάλογα με την χρήση τους, δηλαδή τα request αν και για μικρό αριθμό request είναι δωρεάν.

Μερικά από τα διαθέσιμα API της Google είναι τα παρακάτω:

- **Maps SDK for Android:** Παρέχει γενικές πληροφορίες για τους χάρτες.
- **Places API**: Παρέχει πληροφορίες για ένα μέρος.
- **Directions API**: Παρέχει οδηγίες μεταφοράς από ένα σημείο στον χάρτη σε ένα άλλο.
- **Distance Matrix API**: Παρέχει πληροφορίες όπως η απόσταση και ο χρόνος ταξιδιού ανάμεσα σε δύο σημεία στον χάρτη.
- **Maps Elevation API**: Παρέχει πληροφορίες σχετικά με το υψόμετρο ή το βάθος των ωκεανών.

- **Geocoding API**: Μετατρέπει διευθύνσεις σε γεωγραφικές συντεταγμένες (γεωγραφικό πλάτος και γεωγραφικό μήκος) και το αντίστροφο

Για τις ανάγκες της παρούσας Δ.Ε. έγινε χρήση του Maps SDK for Android και του Directions API.

1.10 Σχεδίαση διεπαφών

Η σχεδίαση των διεπαφών αποτελεί ένα βασικό κομμάτι της ανάπτυξης εφαρμογών. Στο Android Studio οι διεπαφές αναπτύσσονται εύκολα με την χρήση της Xml αλλά και έτοιμων Views που περιέχει το Android Studio.

1.10.1 Σχήματα

Τα συνηθισμένα σχήματα που συναντώνται στις εφαρμογές είναι τα τετράγωνα και τα ορθογώνια παραλληλόγραμμα. Τα σχήματα αυτά δίνουν την αίσθηση στον χρήστη ότι η εφαρμογή είναι αξιόπιστη και ασφαλής. Στην εφαρμογή χρησιμοποιήθηκαν τέτοια σχήματα κυρίως καθώς και ορθογώνια με στρογγυλημένες γωνίες.

Επίσης έχει γίνει χρήση κάποιων μικρών εικονιδίων που βοηθούν στην περιγραφή μιας επιλογής ή μιας κατάστασης, όπως για τα αγαπημένα χρησιμοποιήθηκε το σχήμα του αστεριού ή για το προφίλ του χρήστη το σχήμα ενός προσώπου.

1.10.2 Χρώματα

Τα χρώματα περιγράφουν τα αντικείμενα μέσα σε μία διεπαφή και βοηθούν στην διαφοροποίηση του ενός από του άλλου αντικειμένου. Τα χρώματα επίσης αποτελούν μέρος της ψυχολογίας των ανθρώπων.

Η παρούσα εφαρμογή έχει κατά βάση σαν χρώμα το λευκό καθώς είναι ένα ξεκούραστο στο μάτι χρώμα και είναι πιο εύκολα διακριτό κατά το φως του ήλιου. Τα text views που περιέχουν κείμενο είναι σε μαύρο χρώμα. Τα Buttons είναι μπλέ χρώμα ενώ μέσα στην εφαρμογή υπάρχουν και άλλα ανοικτά χρώματα εκτός από το λευκό όπως είναι στο bottom navigation. Κατά την σύνδεση του χρήστη και την εγγραφή του το κύριο χρώμα των διεπαφών αυτών είναι το πράσινο καθώς αυτό βοηθάει στην συγκέντρωση του χρήστη ώστε να συμπληρώσει σωστά τα στοιχεία του.

1.11 Επίλογος

Στο κεφάλαιο αυτό παρουσιάστηκαν τα συστατικά των τεχνολογιών που χρησιμοποιήθηκαν όπως το Android Studio και αναλύθηκε η αρχιτεκτονική του. Παρουσιαστήκαν τα βασικά δομικά στοιχεία των Android εφαρμογών και αναλύθηκε ο τρόπος λειτουργίας ενός Activity. Επίσης αναφέρθηκαν κάποια χαρακτηριστικά της γλώσσας προγραμματισμού Java η οποία χρησιμοποιήθηκε για την ανάπτυξη της εφαρμογής. Έγινε αναφορά στο Geolocation Android το οποίο χρησιμοποιείται για την εύρεση της τοποθεσίας σε μία συσκευή. Παρουσιάστηκαν κάποιες πληροφορίες για τους χάρτες της Google και τα API που διαθέτει η Google στους προγραμματιστές μέσω του Google Cloud Console και τέλος έγινε μια αναφορά στα βασικά στοιχεία της σχεδίασης διεπαφών όπως σχετικά με τα χρώματα και τα σχήματα που χρησιμοποιήθηκαν.

Στο επόμενο κεφάλαιο θα παρουσιαστούν οι λειτουργίες που παρέχει η εφαρμογή με την χρήση εικόνων για την καλύτερη κατανόηση αυτής από τον αναγνώστη.

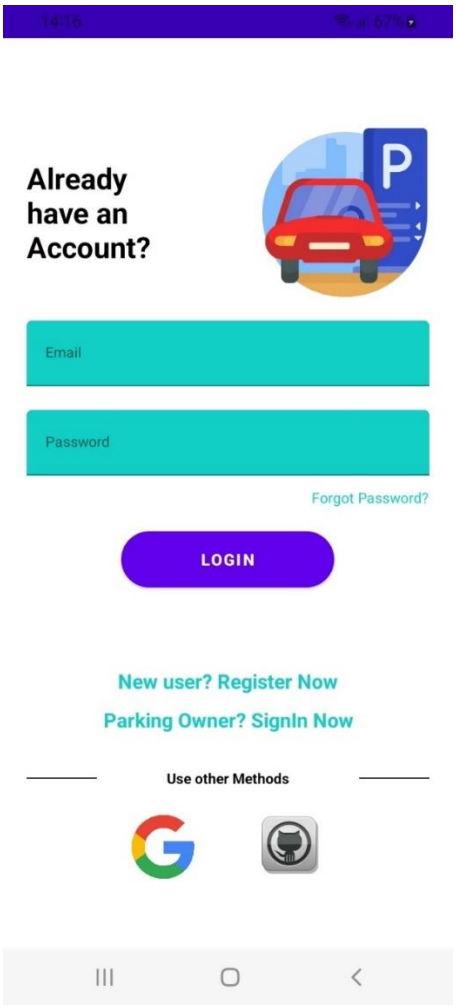
Κεφάλαιο 2ο: Λειτουργίες που παρέχει η εφαρμογή

2.1 Εισαγωγή

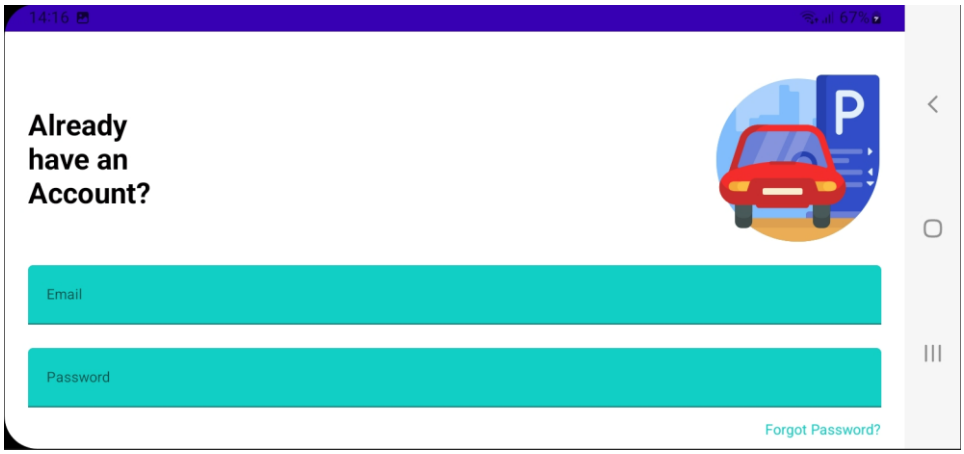
Στο κεφάλαιο αυτό θα παρουσιαστούν οι λειτουργίες που παρέχει η εφαρμογή στον χρήστη καθώς και η διεπαφή χρήστη. Η παρουσίαση των διεπαφών χρήστη θα γίνει και σε portrait αλλά και landscape.

2.2 Σύνδεση του χρήστη

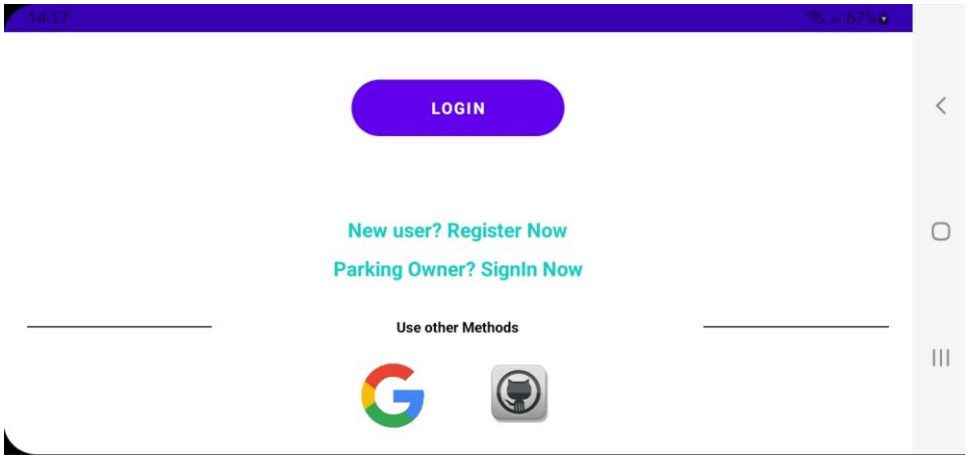
Η πρώτη εικόνα που θα δεί ο χρήστης όταν ανοίξει την εφαρμογή είναι η διεπαφή της σύνδεσης ενός χρήστη στην εφαρμογή. Στην Εικόνα 4 φαίνεται η διεπαφή σύνδεσης του χρήστη στην εφαρμογή σε portrait ενώ στην Εικόνα 5 και στην Εικόνα 6 σε landscape.



Εικόνα 4: Η διεπαφή σύνδεσης στην εφαρμογή portrait



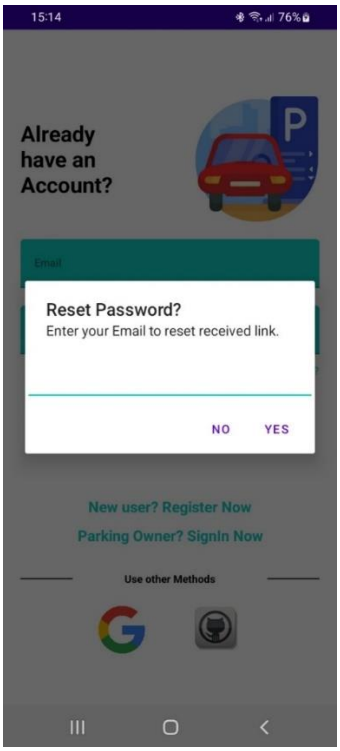
Εικόνα 5: Η διεπαφή σύνδεσης στην εφαρμογή landscape(1)



Εικόνα 6: Η διεπαφή σύνδεσης στην εφαρμογή landscape(2)

Η συγκεκριμένη διεπαφή όπως φαίνεται περιλαμβάνει κάποια views για να μπορεί να αλληλεπιδρά με τον χρήστη. Στο επάνω μέρος της διεπαφής υπάρχει ένα Relative Layout το οποίο περιλαμβάνει ένα Text View με κείμενο όπως φαίνεται στα αριστερά και ένα Image View στα δεξιά. Στη συνέχεια υπάρχουν δύο Edit Text για την συμπλήρωση από τον χρήστη του email του και του κωδικού πρόσβασης. Κάτω από το Edit Text του κωδικού πρόσβασης υπάρχει ένα Text View σε περίπτωση που ο χρήστης ξεχάσει των κωδικό πρόσβασης του. Στη συνέχεια υπάρχει το Button για την σύνδεση του χρήστη στην εφαρμογή και κάτω από το Button αυτό υπάρχουν δύο Text Views, ένα για την εγγραφή του χρήστη στην εφαρμογή και ένα για την σύνδεση του διαχειριστή κάποιου σταθμού στάθμευσης. Επίσης υπάρχει και η δυνατότητα στον χρήστη να συνδεθεί και με άλλους τρόπους στην εφαρμογή εκτός από την χρήση email και password όπως με την χρήση του Google λογαριασμού του ή με τον GitHub λογαριασμό.

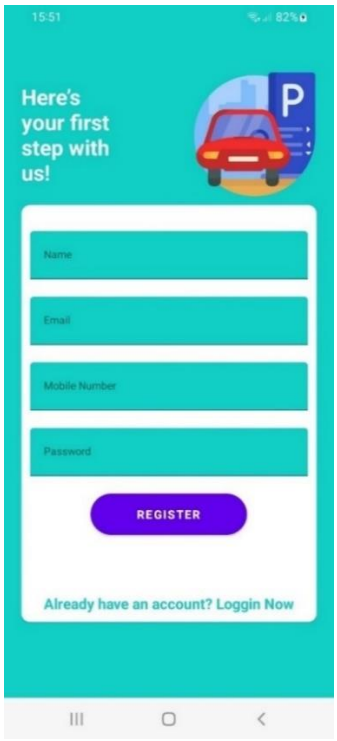
Στην Εικόνα 7 φαίνεται σε portrait το Forgot your password όπου ο χρήστης δίνοντας το email του λαμβάνει link στο email για την επαναφορά του κωδικού πρόσβασης.



Εικόνα 7: Forgot your password portrait

2.3 Εγγραφή του χρήστη στην εφαρμογή

Εάν ένας χρήστης δεν διαθέτει λογαριασμό στην εφαρμογή μπορεί να δημιουργήσει έναν πατώντας το Text View ‘New user? Register Now’ και να μεταφερθεί στο SignUp Activity. Η διεπαφή του SignUp Activity φαίνεται σε portrait και landscape στις Εικόνες 9 και 10.



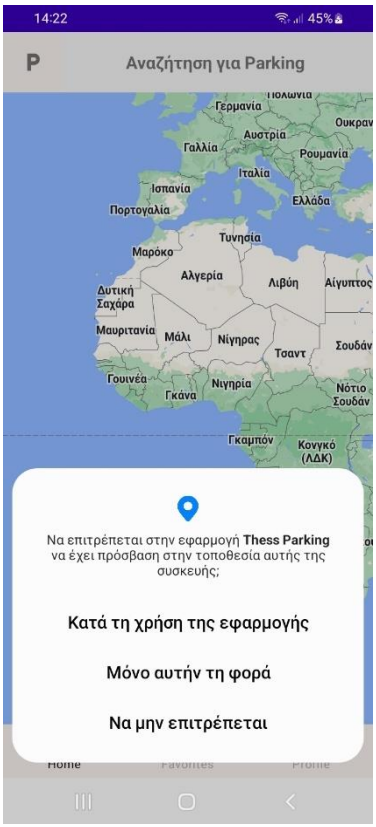
Εικόνα 8: Εγγραφή του χρήστη στην εφαρμογή portrait

Εικόνα 9: Εγγραφή του χρήστη στην εφαρμογή landscape

Όπως φαίνεται και από τις παραπάνω εικόνες η διεπαφή της δημιουργίας λογαριασμού του χρήστη αποτελείται από ένα Relative Layout όπου στα αριστερά υπάρχει ένα Text View και στα δεξιά ένα Image View. Κάτω από το Relative Layout υπάρχουν τέσσερα Edit Texts όπου ο χρήστης συμπληρώνει τα στοιχεία του, ένα Button για να κάνει εγγραφή στην εφαρμογή και ένα Text View όπου πατώντας το ο χρήστης επιστρέφει στην προηγούμενη διεπαφή της σύνδεσης στην εφαρμογή.

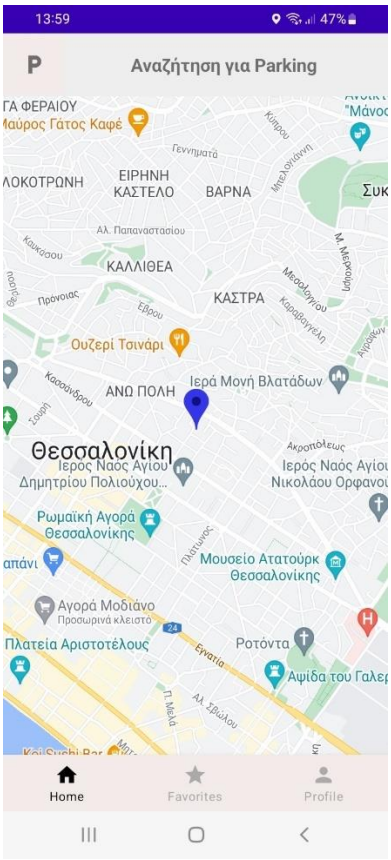
2.4 Η Εφαρμογή μετά την εισόδου του χρήστη

Μετά την επιτυχή σύνδεση του χρήστη στην εφαρμογή η πρώτη εικόνα που εμφανίζεται είναι η αίτηση άδειας χρήσης της τοποθεσίας όπως φαίνεται στην Εικόνα 10.

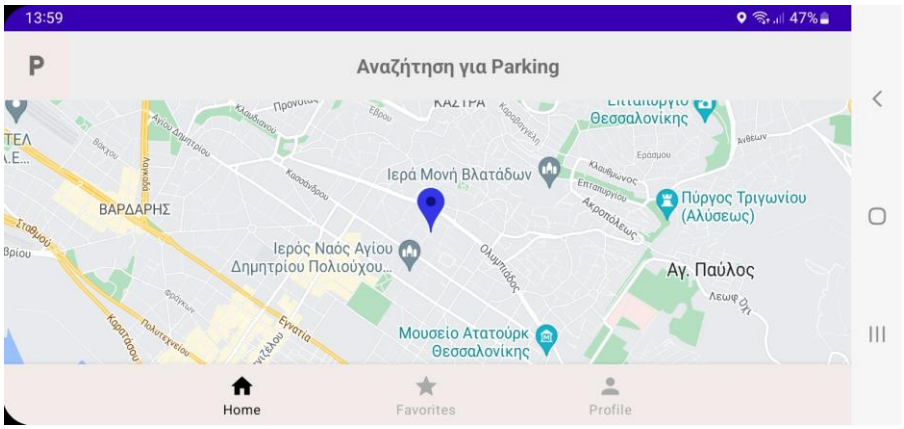


Εικόνα 10: Αίτηση άδειας χρήσης της τοποθεσίας του χρήστη

Αφού ο χρήστης δεχθεί την αίτηση εμφανίζεται με μπλέ marker (πινέζα) η τοποθεσία του όπως φαίνεται στην Εικόνα 11 και 12.



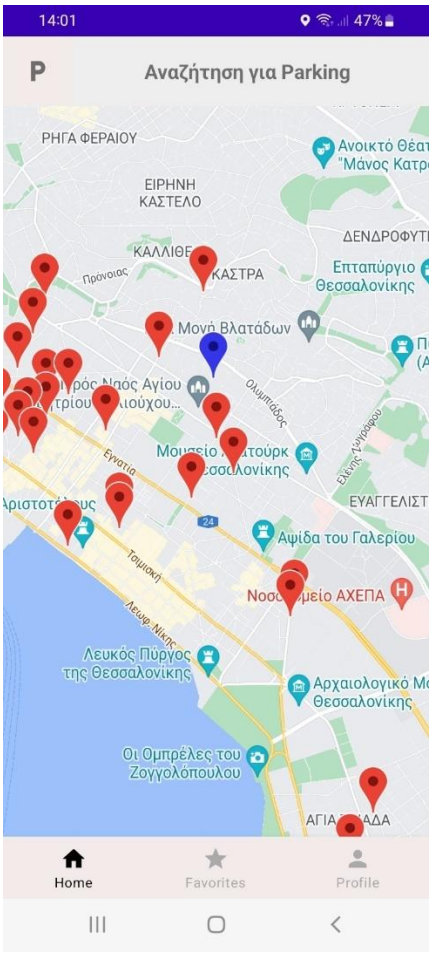
Εικόνα 11: Τρέχων τοποθεσία του χρήστη portrait



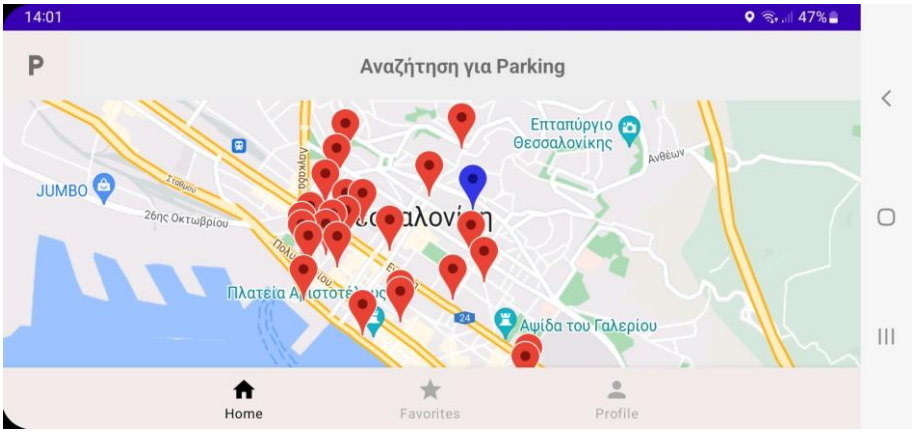
Εικόνα 12: Τρέχων τοποθεσία του χρήστη landscape

Όπως φαίνεται από τις παραπάνω εικόνες υπάρχει ένα bottom navigation για να μπορεί ο χρήστης ανά πάσα στιγμή να μετακινείται από τους χάρτες στα αγαπημένα του η στο προφίλ του.

Η αρχική οθόνη περιλαμβάνει έναν χάρτη Google και ένα Image Button επάνω δεξιά το οποίο όταν πατηθεί εμφανίζει στον χρήστη τους σταθμούς στάθμευσης με την μορφή κόκκινων πινεζών στον χάρτη όπως φαίνεται παρακάτω:



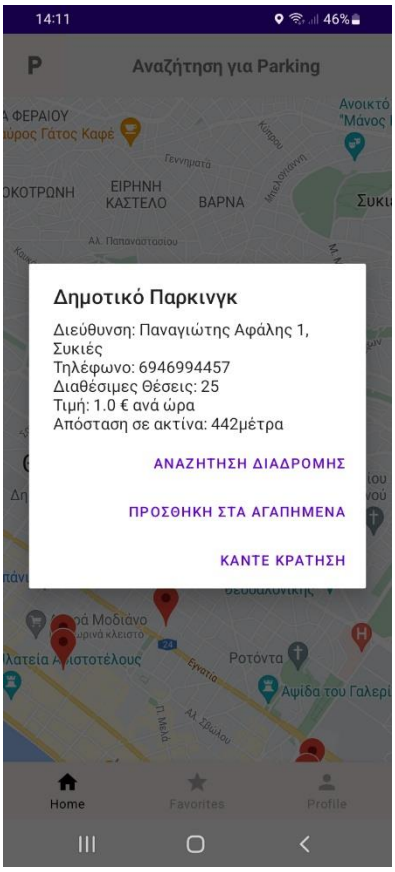
Εικόνα 13: Σταθμοί στάθμευσης στον χάρτη portrait



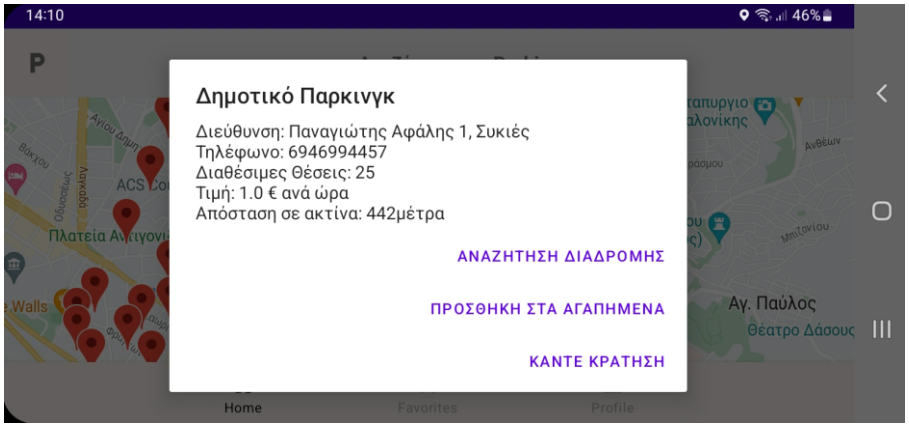
Εικόνα 14: Σταθμοί στάθμευσης στον χάρτη landscape

Έτσι ο χρήστης μπορεί να διαλέξει έναν σταθμό στάθμευσης που τον εξυπηρετεί πατώντας πάνω στον market. Πατώντας στον marker εμφανίζονται στον χρήστη πληροφορίες σχετικά με τον συγκεκριμένο σταθμό στάθμευσης σε μορφή Alert Dialog καθώς και τρεις επιλογές, της αναζήτησης διαδρομής από την τρέχουσα τοποθεσία του στον επιλεγμένο σταθμό, της αποθήκευσης στα αγαπημένα όπου ο συγκεκριμένος σταθμός στάθμευσης αποθηκεύεται στη βάση δεδομένων στα αγαπημένα του χρήστη και η επιλογή της κράτησης όπου ο χρήστης μπορεί να κάνει κράτηση θέσης σε έναν σταθμό ώστε να είναι σίγουρος ότι θα βρει να παρκάρει.

Οι πληροφορίες για τον σταθμό που εμφανίζονται στον χρήστη είναι η διεύθυνση του σταθμού, το τηλέφωνο του, οι διαθέσιμες θέσεις που έχει ο σταθμός, η τιμή στάθμευσης ανά ώρα καθώς και η απόσταση σε ακτίνα από την τοποθεσία του έως αυτόν όπως φαίνεται στην Εικόνα 15 και 16.



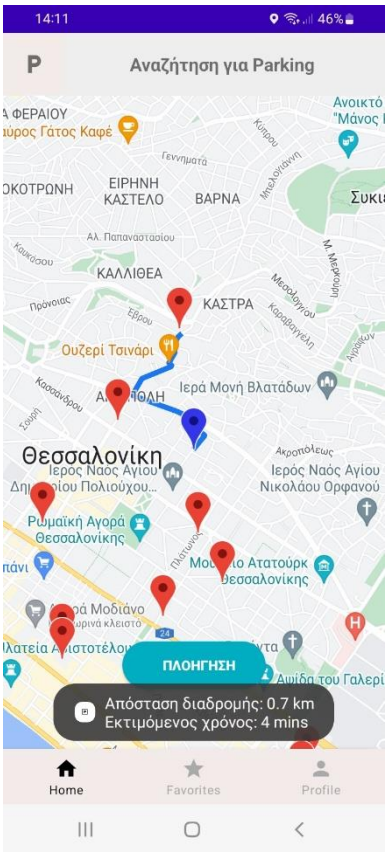
Εικόνα 15: Πληροφορίες Σταθμού portrait



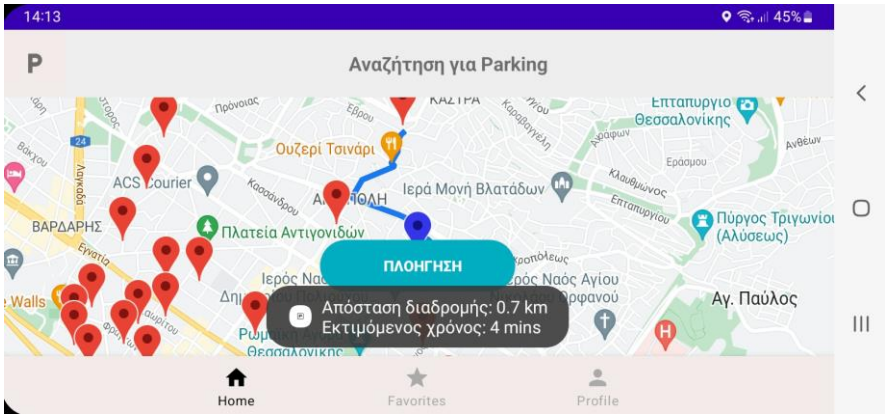
Εικόνα 16: Πληροφορίες Σταθμού landscape

2.4.1 Αναζήτηση Διαδρομής

Όταν ο χρήστης επιλέξει την αναζήτηση διαδρομής τότε εμφανίζεται στον χάρτη η συντομότερη διαδρομή ανάμεσα στην τοποθεσία του και στον σταθμό που έχει επιλέξει καθώς και ένα μήνυμα σε μορφή Toast που του εμφανίζει την απόσταση σε χιλιόμετρα καθώς και τον εκτιμώμενο χρόνο όπως φαίνεται παρακάτω:

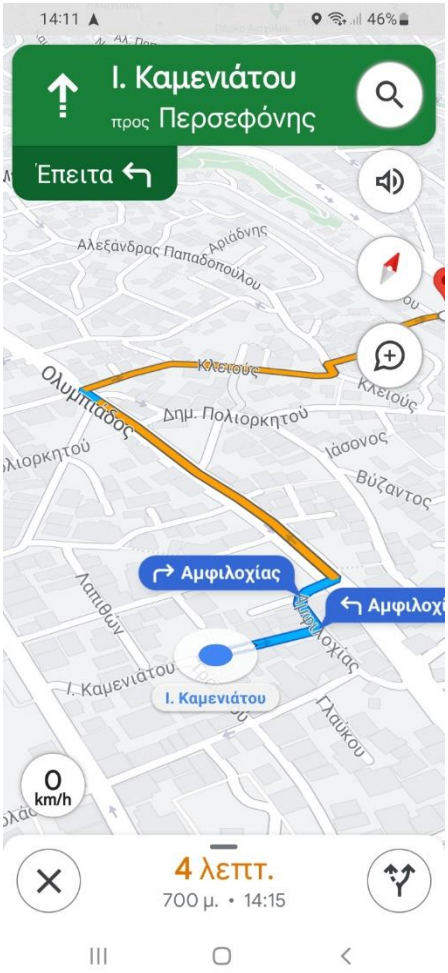


Εικόνα 17: Συντομότερη διαδρομή portrait



Εικόνα 18: Συντομότερη διαδρομή landscape

Επίσης εμφανίζεται και το Button Πλοήγηση το οποίο όταν πατηθεί ανοίγει το GPS του Google Maps όπως φαίνεται στην Εικόνα 19:



Εικόνα 19: Πλοήγηση μέσω Google Maps

Ο τρόπος εύρεσης της συντομότερης διαδρομής και η εμφάνιση polyline στον χάρτη έγινε με την χρήση του Direction API της Google και θα παρουσιαστεί αναλυτικά στο επόμενο κεφάλαιο.

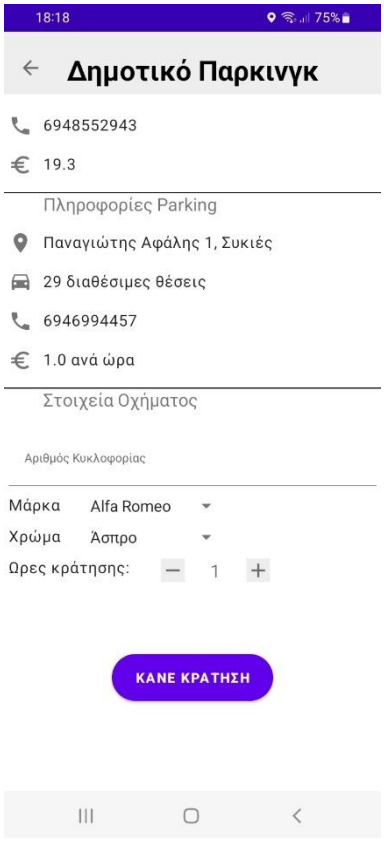
2.4.2 Προσθήκη στα αγαπημένα

Όταν ο χρήστης επιλέξει το προσθήκη στα αγαπημένα τα στοιχεία του σταθμού στάθμευσης αποθηκεύονται στην Realtime Database της Firebase. Εάν η αποθήκευση των δεδομένων είναι επιτυχής θα εμφανιστεί στον χρήστη ένα μήνυμα σε μορφή Toast ότι η έγινε αποθήκευση στα αγαπημένα ενώ εάν η αποθήκευση είναι ανεπιτυχής θα εμφανιστεί μήνυμα στον χρήστη ότι κάτι πήγε στραβά.

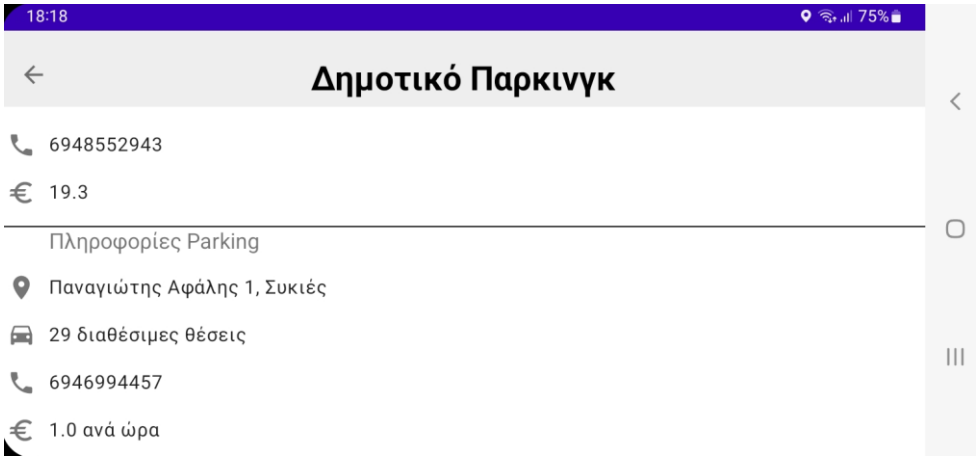
2.4.3 Κράτηση θέσης στάθμευσης

Αν ο χρήστης επιλέξει να κάνει κράτηση τότε εμφανίζεται το Booking Activity στην οθόνη του χρήστη. Το Booking Activity όπως φαίνεται και στις παρακάτω εικόνες αποτελείται στην κορυφή από ένα Relative Layout που περιέχει ένα Image Button στα αριστερά και ένα Text View στο κέντρο. Στο Text View εμφανίζεται η επωνυμία του σταθμού στάθμευσης ενώ ο χρήστης πατώντας το Image Button επιστρέφει στους χάρτες. Στη συνέχεια υπάρχει ένα ακόμα Relative Layout στο οποίο υπάρχουν Image Views και Text Views όπου εμφανίζονται σχετικά με τον χρήστη ο αριθμός τηλεφώνου του και το διαθέσιμο υπόλοιπο του. Εδώ να αναφέρουμε ότι το υπόλοιπο είναι εικονικό. Όταν ένας χρήστης κάνει εγγραφή στην εφαρμογή ξεκινάει με το ποσό των 30 ευρώ. Σε μελλοντική επέκταση της εφαρμογής ο χρήστης μέσω χρεωστικής κάρτας θα μπορεί να προσθέσει χρήματα στον λογαριασμό του. Επίσης

εμφανίζονται και οι πληροφορίες του σταθμού στάθμευσης όπως η διεύθυνση, οι διαθέσιμες θέσεις, το τηλέφωνο και η χρέωση ανά ώρα στάθμευσης. Ο χρήστης θα πρέπει να εισάγει των αριθμό κυκλοφορίας του οχήματος του σε ένα ένα Edit Text καθώς να επιλέξει την μάρκα και το χρώμα του οχήματος του μέσα από ένα spinner αντίστοιχα και να ορίσει για πόσες ώρες θα είναι η κράτηση του. Οι ώρες κράτησεις εμφανίζονται σε ένα Text View και μπορεί να τις αυξομειώσει με την χρήση των δύο Image Buttons – και +.

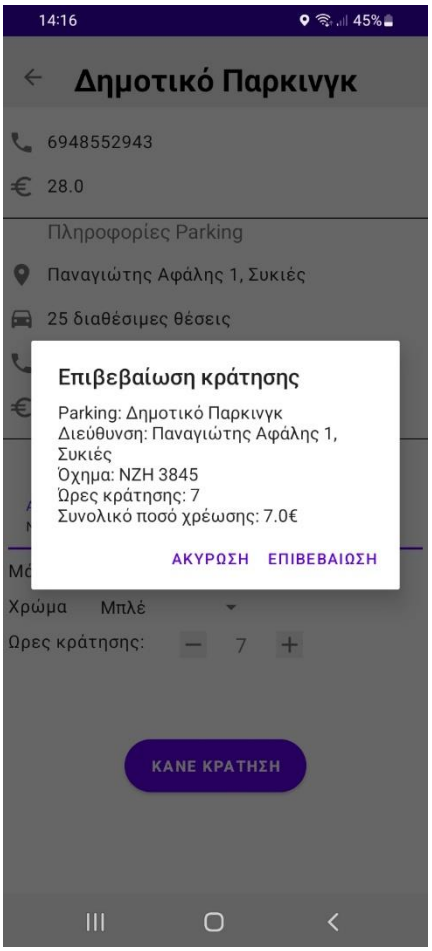


Εικόνα 20: Διεπαφή Κρατήσης θέσεων σταθμευσης portrait



Εικόνα 21: Διεπαφή Κράτησης θέσεων στάθμευσης landscape

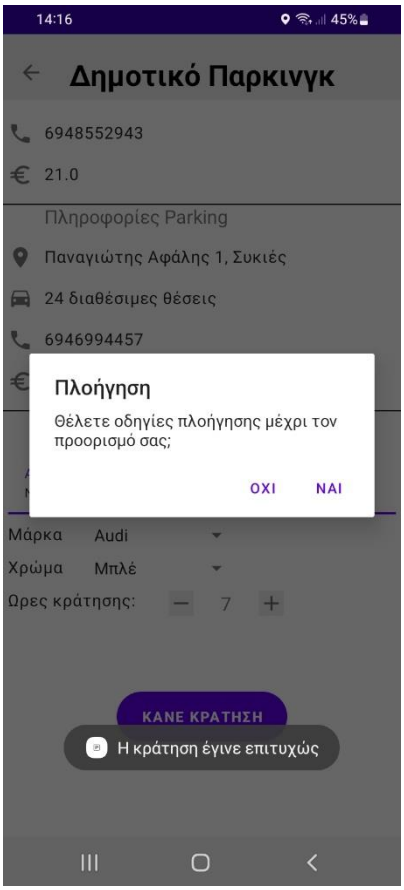
Εφόσον ο χρήστης συμπληρώσει όλα τα στοιχεία, επιλέξει τις ώρες κράτησης και πατήσει το Button Κάνε κράτηση εμφανίζεται ένα Alert Dialog που περιέχει το όνομα του σταθμού στάθμευσης, την διεύθυνση του, το σύνολο των ωρών που έχει επιλέξει όπως φαίνεται στην Εικόνα 22.



Εικόνα 22: Επιβεβαίωση Κράτησης

Εάν ο χρήστης επιλέξει ακύρωση το Alert Dialog κλείνει, ενώ εάν επιλέξει επιβεβαίωση κράτησης αρχικά γίνεται έλεγχος εάν το διαθέσιμο ποσό του επαρκεί για την κράτηση και αν υπάρχουν διαθέσιμες θέσεις. Εάν όχι εμφανίζεται το μήνυμα Το υπόλοιπο σου δεν επαρκεί και το Δεν υπάρχουν διαθέσιμες θέσεις στάθμευσης αντίστοιχα.

Εάν ο έλεγχος είναι επιτυχής τότε γίνεται αφαίρεση του συνολικού κόστους από το διαθέσιμο ποσό του χρήστη και οι διαθέσιμες θέσεις του σταθμού μειώνονται κατά ένα. Τα νέα δεδομένα για τις διαθέσιμες θέσεις του σταθμού και το διαθέσιμο χρηματικό ποσό του χρήστη αναβαθμίζονται στην βάση δεδομένων καθώς αποθηκεύεται και η κράτηση στην Realtime Database της Firestore. Επίσης εμφανίζεται ένα μήνυμα σε μορφή Toast ότι η κράτηση έγινε επιτυχώς και εμφανίζει ένα Alert Dialog για το εάν ο χρήστης θέλει οδηγίες πλοήγησης για τον σταθμό που έχει επιλέξει όπως φαίνεται στην Εικόνα 23.

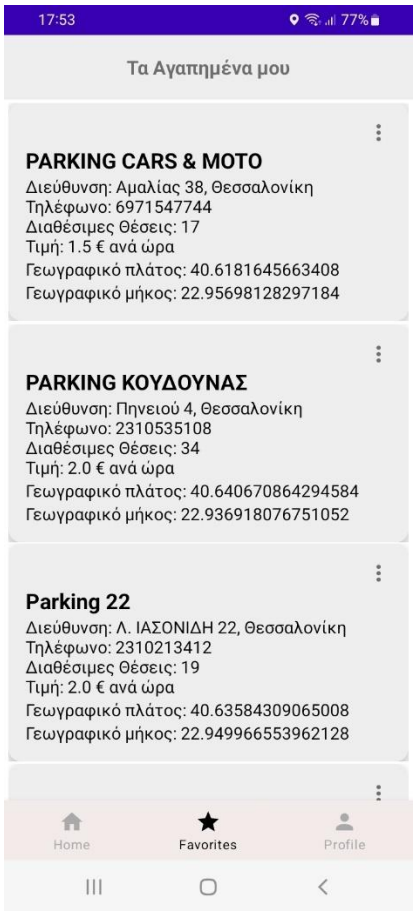


Εικόνα 23: Ερώτηση του χρήστη αν επιθυμεί πλοήγηση

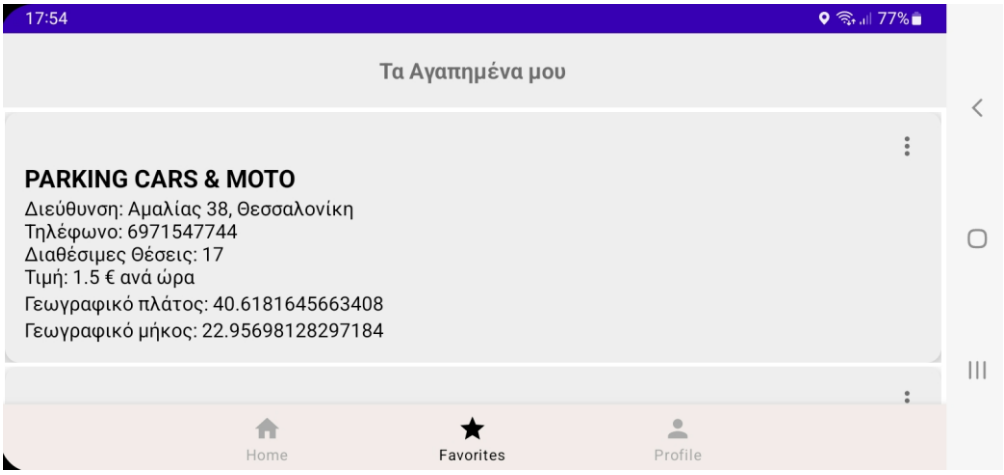
Αν ο χρήστης επιλέξει Όχι το Alert Dialog κλείνει ενώ εάν επιλέξει Ναι τότε ανοίγει το GPS του Google Maps.

2.5 Τα αγαπημένα χρήστη

Επιλέγοντας τα Favorites από το bottom navigation μεταφερόμαστε στην διεπαφή των αγαπημένων του χρήστη. Η διεπαφή αυτή όπως φαίνεται στις Εικόνες 24 και 25 αποτελείται από ένα Relative Layout το οποίο περιέχει ένα Text View με το κείμενο Τα αγαπημένα μου και ένα δεύτερο Relative Layout το οποίο περιέχει ένα scroll view και ένα recycler view που εμφανίζει τους αγαπημένους σταθμούς του χρήστη σε μορφή card view. Για το recycler view θα γίνει ανάλυση στο επόμενο κεφάλαιο.



Εικόνα 24: Τα αγαπημένα του χρήστη portrait

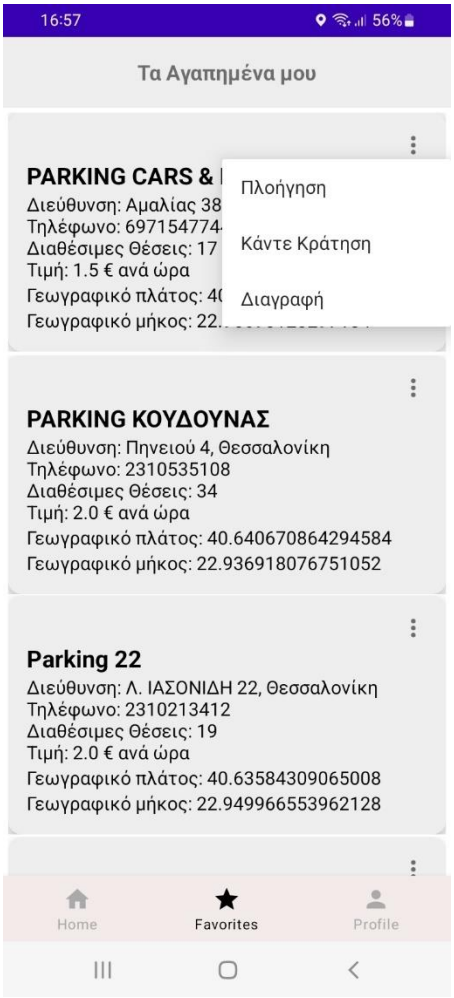


Εικόνα 25: Τα αγαπημένα του χρήστη landscape

Επίσης σε κάθε card αγαπημένων σταθμών υπάρχει ένα Image Button το οποίο περιέχει ένα popup menu όπως φαίνεται στην Εικόνα 26 το οποίο περιέχει τρεις επιλογές.

Η πρώτη επιλογή είναι η πλοήγηση όπου ανοίγει το navigation του Google Maps και δίνει οδηγίες πλοήγησης στον χρήστη από την τρέχουσα τοποθεσία του προς τον επιλεγμένο σταθμό. Η δεύτερη επιλογή είναι αυτή της κράτησης θέσης στον συγκεκριμένο σταθμό και μεταφέρει τον χρήστη στην διεπαφή των κρατήσεων. Η Τρίτη και τελευταία είναι αυτή της διαγραφής του σταθμού από τα

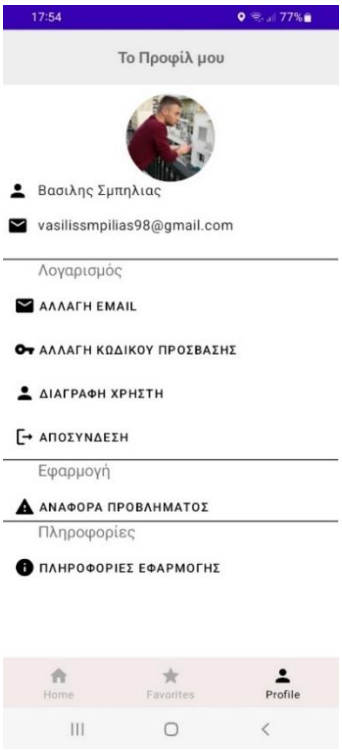
αγαπημένα όπου επιλέγοντας το ο χρήστης διαγράφει τον συγκεκριμένο σταθμό από τη συλλογή των αγαπημένων στην Realtime Database της Firestore.



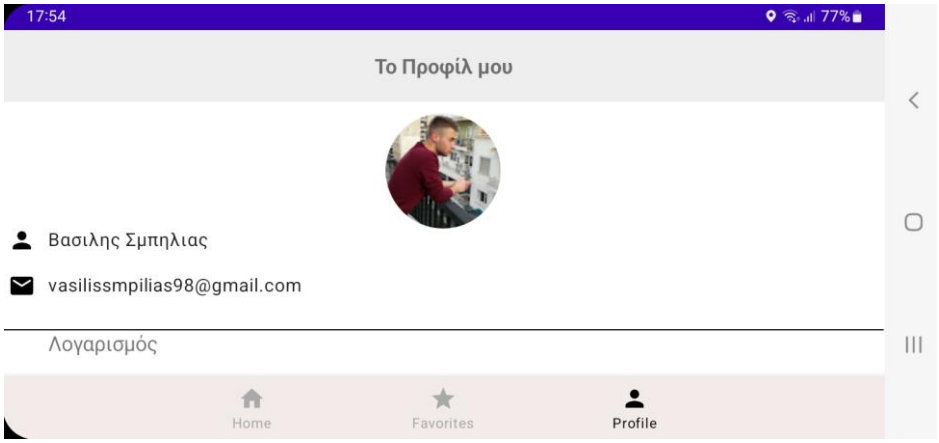
Εικόνα 26: Popup menu αγαπημένων σταθμών

2.6 Το προφίλ του χρήστη

Όταν ο χρήστης επιλέξει το Profile από το bottom navigation τότε μεταφέρεται στο προφίλ του. Η συγκεκριμένη διεπαφή όπως φαίνεται από τις παρακάτω εικόνες αποτελείται από ένα Image View στο οποίο εάν ο χρήστης είναι συνδεδεδεμένος με Google ή Github εμφανίζει την φωτογραφία προφίλ του αλλιώς εμφανίζει ένα black person. Κάτω από το Image View υπάρχουν δύο μικρότερα Image View και δύο Text View τα οποία εμφανίζουν το όνομα του χρήστη και το email του. Στη συνέχεια για την διαχείριση του λογαριασμού υπάρχουν τέσσερα Material Buttons τα οποία στα αριστερά έχουν ένα εικονίδιο και δίπλα από αυτό κείμενο. Τα τέσσερα αυτά Material Buttons είναι η αλλαγή email λογαριασμού, η αλλαγή κωδικού πρόσβασης του λογαριασμού, η διαγραφή του λογαριασμού και η αποσύνδεση του χρήστη. Επίσης υπάρχει ένα Material Button για την αναφορά προβλήματος και ένα Material Button για πληροφορίες σχετικά με την εφαρμογή.



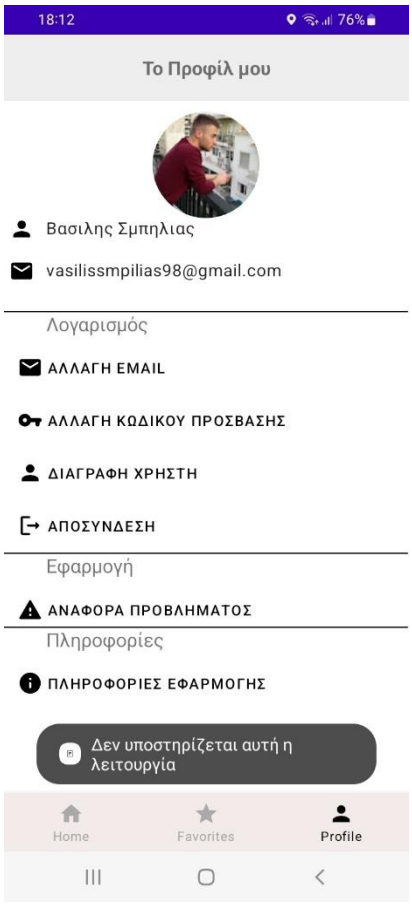
Εικόνα 27: Το προφίλ του χρήστη portrait



Εικόνα 28: Το προφίλ του χρήστη landscape

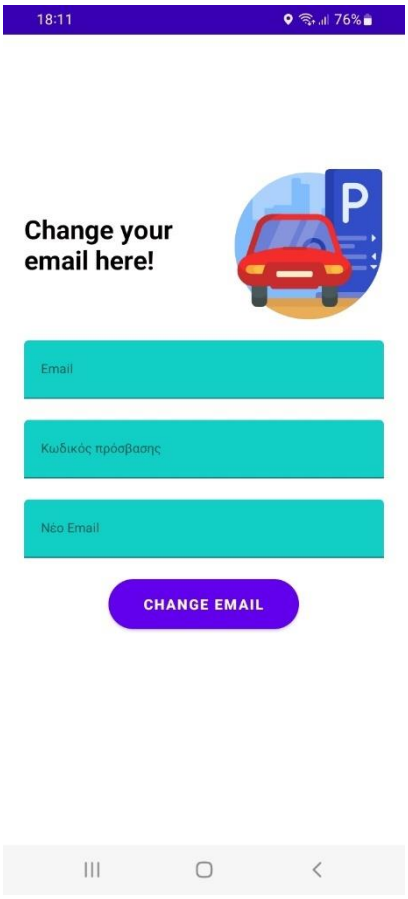
2.6.1 Αλλαγή email και κωδικού πρόσβασης

Υπάρχει δυνατότητα στον χρήστη να αλλάξει τον κωδικό πρόσβασης του ή το email του. Αυτή η δυνατότητα παρέχεται στον χρήστη εφόσον είναι συνδεδεμένος με τον λογαριασμό που έχει δημιουργήσει στην εφαρμογή. Όπως φαίνεται και στην Εικόνα 29 αν ο χρήστης είναι συνδεδεμένος με τον λογαριασμό Google ή Github δεν του επιτρέπει να προχωρήσει στις αλλαγές εμφανίζοντας ένα μήνυμα σε μορφή Toast ότι η λειτουργία αυτή δεν υποστηρίζεται

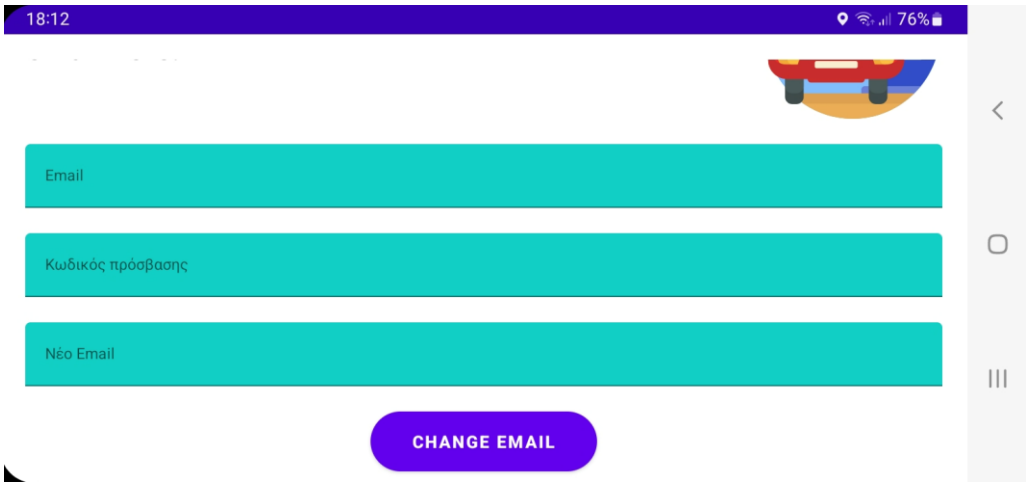


Εικόνα 29: Άρνηση αλλαγής στοιχείων χρήστη

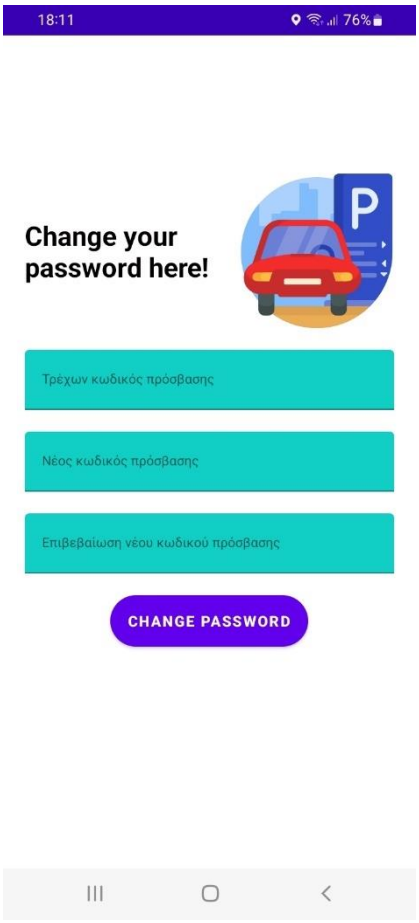
Εφόσον ο χρήστης είναι συνδεδεμένος κανονικά με τον λογαριασμό του πατώντας το Button αλλαγή email μεταφέρεται στην διεπαφή της αλλαγής email ενώ πατώντας το Button αλλαγή κωδικού πρόσβασης μεταφέρεται στην αντίστοιχη διεπαφή. Οι διεπαφές αυτές παρουσιάζονται σε portrait και landscape στις Εικόνες 30, 31 και 32, 33 αντίστοιχα.



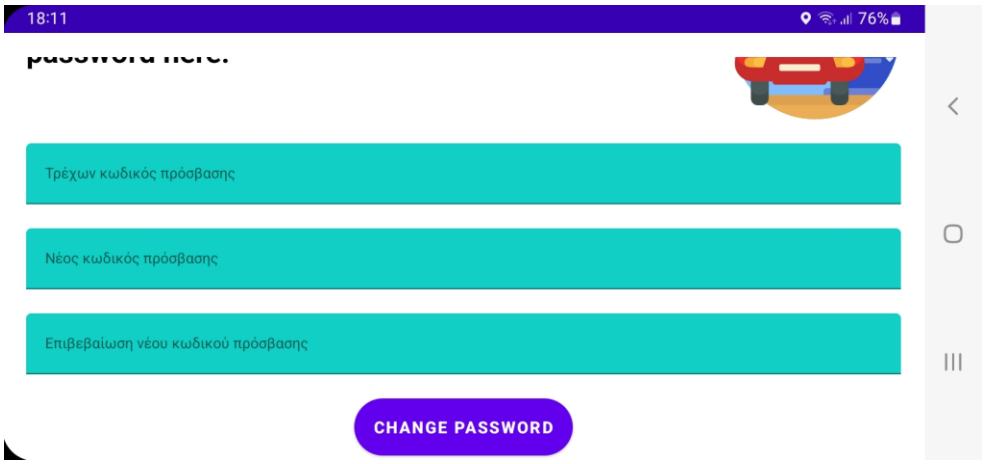
Εικόνα 30: Διεπαφή Αλλαγής Email portrait



Εικόνα 31: Διεπαφή Αλλαγής Email landscape



Εικόνα 32: Διεπαφή Αλλαγής κωδικού πρόσβασης portrait



Εικόνα 33: Διεπαφή Αλλαγής κωδικού πρόσβασης landscape

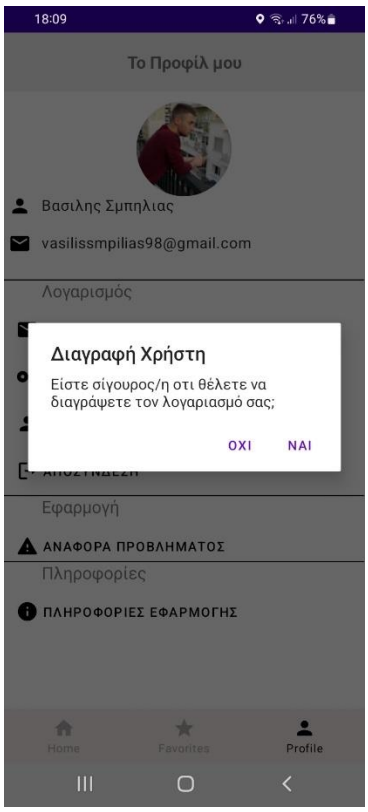
Όπως φαίνεται από τις παραπάνω εικόνες και οι δύο διεπαφές αποτελούνται από ένα Text View στα αριστερά και ένα Image View στα δεξιά. Ακολουθούν 3 Edit Texts και ένα Button.

Για την διεπαφή της αλλαγής email ο χρήστης πρέπει στα Edit Texts να συμπληρώσει το email του τον κωδικό πρόσβασης και το νέο email που θέλει. Πατώντας το Button change password αφού γίνει έλεγχος ότι τα πεδία είναι συμπληρωμένα και το email και το password είναι σωστά τότε γίνεται reauthentication του χρήστη στην firebase. Εφόσον γίνει επιτυχές το reauthentication εμφανίζεται ένα μήνυμα σε μορφή toast και ο χρήστης μεταφέρεται στην διεπαφή της σύνδεσης στον λογαριασμό.

Αντίστοιχα γίνεται και στην διεπαφή της αλλαγής κωδικού πρόσβασης ο χρήστης συμπληρώνει τον κωδικό του, συμπληρώνει τον νέο κωδικό της επιλογής και τον επιβεβαιώνει στο τριτο Edit Text. Αν τα πεδία είναι κατάλληλα γίνεται reauthentication στην firebase και εμφανίζεται στον χρήστη μήνυμα επιτυχής αλλαγής κωδικού πρόσβασης και μεταφέρεται στην διεπαφή της σύνδεσης στην εφαρμογή.

2.6.2 Διαγραφή Χρήστη

Εάν ο χρήστης το επιθυμεί έχει την δυνατότητα να διαγράψει τον λογαριασμό του πατώντας το Material Button Διαγραφή Χρήστη. Όταν ο χρήστης το πατήσει όπως φαίνεται και στην Εικόνα 34 εμφανίζεται ένα Alert Dialog που ρωτάει τον χρήστη εάν είναι σίγουρος ότι θέλει να διαγράψει τον λογαριασμό του.

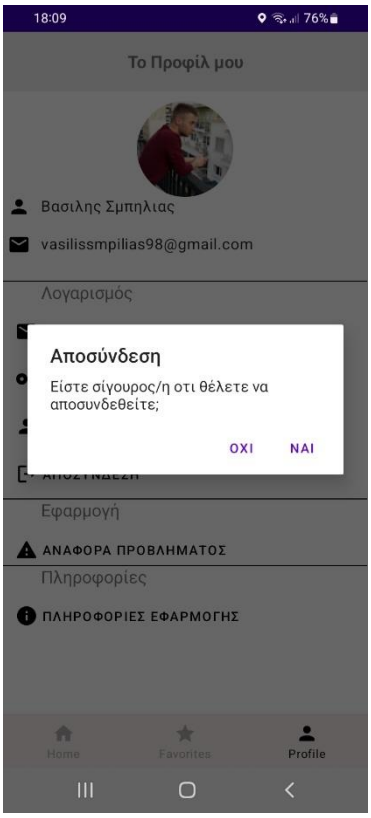


Εικόνα 34: Προειδοποίηση για διαγραφή του λογαριασμού του

Αν ο χρήστης πατήσει Όχι το Dialog κλείνει ενώ αν πατήσει Ναι διαγράφεται ο λογαριασμός του από τους Users και τα επιμέρους στοιχεία του που υπάρχουν στην συλλογή User_Info της Firestore. Εάν η διαγραφή είναι επιτυχής θα εμφανιστεί στον χρήστη μήνυμα σε μορφή Toast και θα μεταφερθεί στην διεπαφή σύνδεσης στην εφαρμογή.

2.6.3 Αποσύνδεση Χρήστη

Παρόμοια με την Διαγραφή χρήστη είναι και η αποσύνδεση του. Αν ο χρήστης πατήσει το Material Button Αποσύνδεση εμφανίζεται όπως φαίνεται και στην Εικόνα 35 ένα Alert Dialog που ρωτάει τον χρήστη αν είναι σίγουρος ότι θέλει να αποσυνδεθεί από την εφαρμογή.



Εικόνα 35: Προειδοποίηση για αποσύνδεση

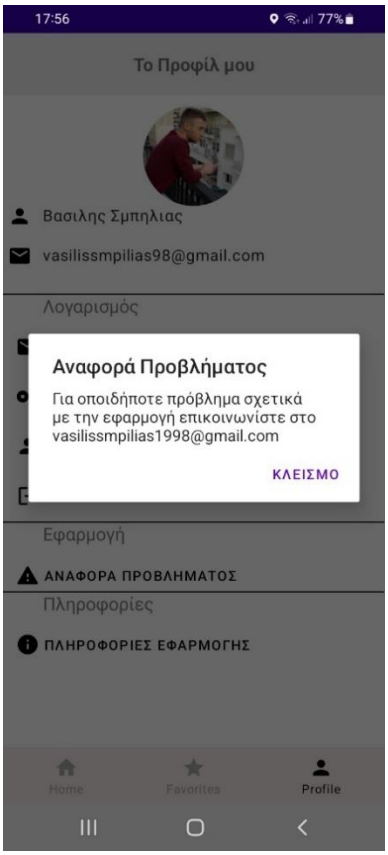
Εάν ο χρήστης πατήσει Όχι το Alert Dialog κλείνει ενώ εάν πατήσει Ναι γίνεται αποσύνδεση από την firebase και ο χρήστης μεταφέρεται στην διεπαφή της σύνδεσης στην εφαρμογή.

2.6.4 Αναφορά προβλήματος & Πληροφορίες εφαρμογής

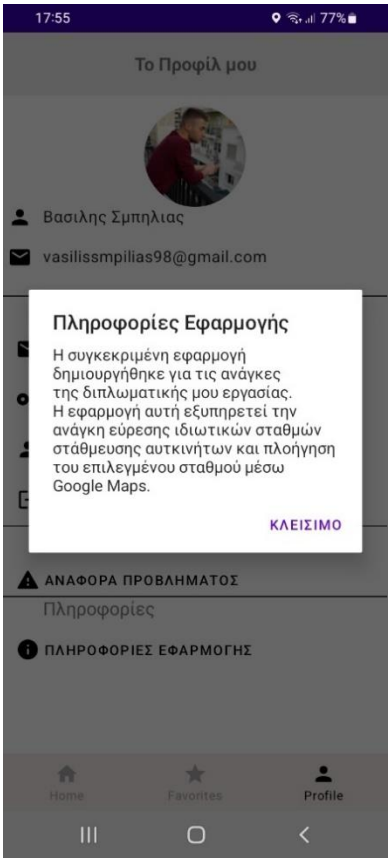
Το Material Button Αναφορά προβλήματος όταν πατηθεί εμφανίζει ένα Alert Dialog που λέει στον χρηστη να επικοινωνήσει για οποιοδήποτε πρόβλημα με τον διαχειριστή της εφαρμογής.

Παρόμοια το Material Button Πληροφορίες εφαρμογής όταν πατηθεί εμφανίζει ένα Alert Dialog που αναφέρει στον χρηστη δύο λόγια για την εφαρμογή και τον λόγο που δημιουργήθηκε.

Στις Εικόνες 36 και 37 φαίνονται τα Alert Dialogs που εμφανίζονται στον χρήστη για την αναφορά προβλήματος και τις πληροφορίες εφαρμογής αντίστοιχα.



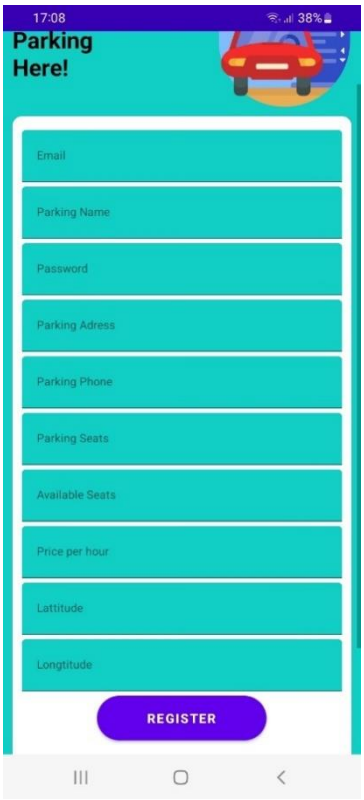
Εικόνα 36: Αναφορά Προβληματος



Εικόνα 37: Πληροφορίες εφαρμογής

2.7 Διαχειριστής των σταθμών στάθμευσης

Για την διαχείριση των σταθμών στάθμευσης στην εφαρμογή υπάρχει η δυνατότητα για τον ιδιοκτήτη του σταθμού ή κάποιον διαχειριστή του να επεξεργάζεται και να αναβαθμίζει τις πληροφορίες του σταθμού. Ο διαχειριστής του σταθμού μπορεί να συνδεθεί στην εφαρμογή στην διεπαφή του διαχειριστή πατώντας στο Text View 'Parking Owner? Sign In now' όπως φαίνεται στην Εικόνα 4. Πατώντας στο συγκεκριμένο Text View ο διαχειριστής μεταφέρεται στην διεπαφή σύνδεσης του διαχειριστή η οποία είναι ίδια με την διεπαφή σύνδεσης του χρήστη στην εφαρμογή μόνο που δεν διαθέτει την δυνατότητα σύνδεσης με Google account ή Github. Εφόσον δεν υπάρχει λογαριασμός ο διαχειριστής μπορεί να κάνει εγγραφή τον σταθμό του στην εφαρμογή πατώντας το Text View 'New Parking Owner? Register now. Η διεπαφή της εγγραφής ενός σταθμού στην εφαρμογή φαίνεται σε portrait στην Εικόνα 38.

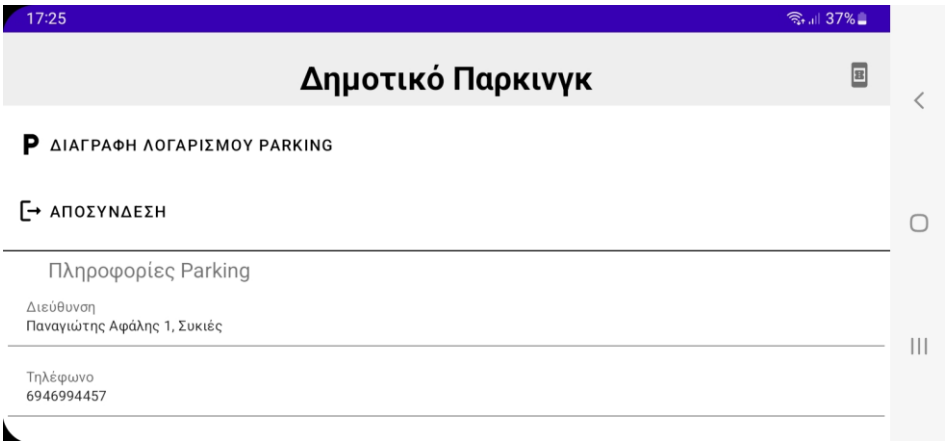


Εικόνα 38: Εγγραφή του σταθμού στην εφαρμογή portrait

Η διεπαφή αυτή αποτελείται από ένα Text View στα αριστερά και ένα Image View στα δεξιά. Ακολουθούν δέκα Edit Text στα οποία ο διαχειριστής πρέπει να συμπληρώσει τα στοιχεία για τον σταθμό στάθμευσης. Τα πεδία που πρέπει να συμπληρώσει ο διαχειριστής για την εγγραφή του σταθμού είναι το email του σταθμού, ο κωδικός πρόσβασης για είσοδο στην εφαρμογή, το όνομα του σταθμού, την διεύθυνση του σταθμού, το τηλέφωνο του σταθερό ή κινητό, τις συνολικές θέσεις στάθμευσης που διαθέτει ο σταθμός, τις διαθέσιμες θέσεις του, το κόστος στάθμευσης ανά ώρα καθώς και τις συντεταγμένες του σταθμού(γεωγραφικό πλάτος και γεωγραφικό μήκος). Αφού συμπληρώσει όλα τα πεδία πατώντας το Button Register δημιουργείται ο λογαριασμός στην firebase και γίνεται η εγγραφή του σταθμού στην Realtime Database.

Εφόσον η σύνδεση ή η εγγραφή του σταθμού στην εφαρμογή γίνει επιτυχώς ο διαχειριστής μεταφέρεται στην διεπαφή του σταθμού όπως φαίνεται στις παρακάτω Εικόνες.

Εικόνα 39: Προφίλ σταθμού portrait



Εικόνα 40: Προφίλ σταθμού landscape

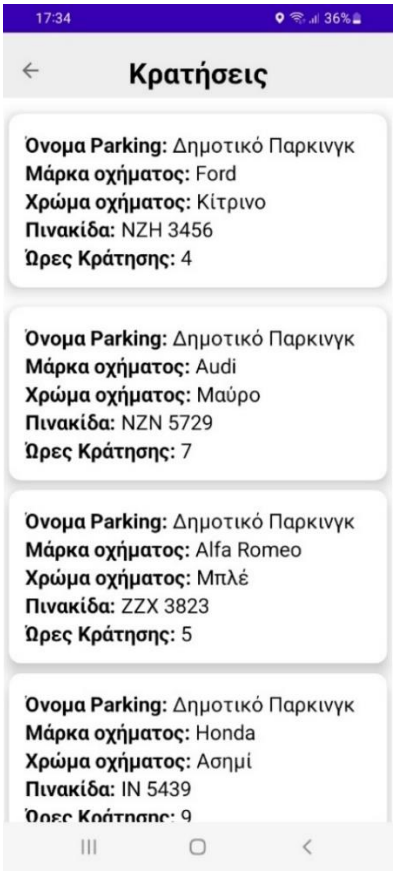
Στο προφίλ του σταθμού ο διαχειριστής μπορεί να βλέπει τις πληροφορίες σχετικά με τον σταθμό και να τις επεξεργάζεται.

Η διεπαφή αυτή αποτελείται από δύο Material Button, ένα για την διαγραφή του λογαριασμού και του σταθμού από την εφαρμογή και ένα για την αποσύνδεση του διαχειριστή από την εφαρμογή. Όπως και στο προφίλ του χρήστη όταν πατηθεί κάποιο από τα δύο αυτά Material Button εμφανίζεται ένα Alert Dialog που προειδοποιεί τον διαχειριστή εάν θέλει να προβεί στην συγκεκριμένη ενέργεια. Στη συνέχεια υπάρχουν έξι Edit Text όπου η διαχειριστής μπορεί να δεί πληροφορίες του σταθμού όπως την διεύθυνση του σταθμού, το τηλέφωνο του, τις συνολικές θέσεις στάθμευσης κ.α. και να τις διαχειριστεί. Επίσης μπορεί να δεί μέσω ενός Text View τις διαθέσιμες θέσεις στάθμευσης του σταθμού και να τις αυξήσει πατώντας το δεξί Image Button ή να τις μειώσει πατώντας το αριστερό Image Button. Πατώντας το Button Αποθήκευση Αλλαγών γίνεται update των τιμών των πεδίων του σταθμού στην

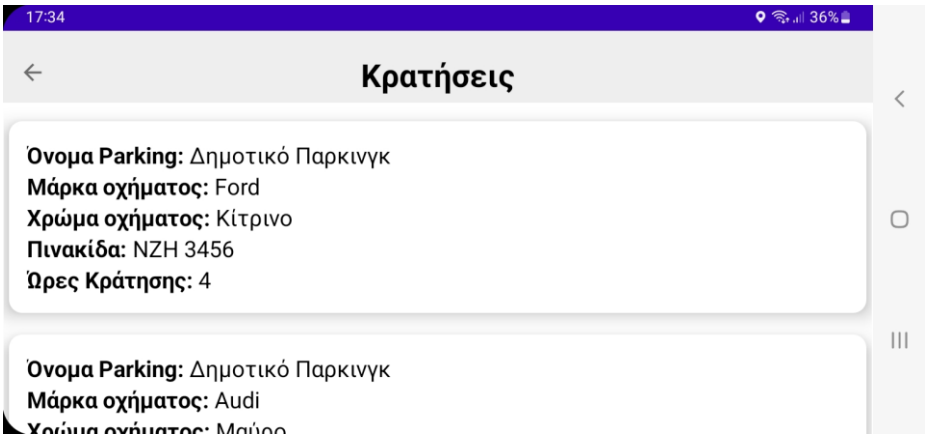
Realtime Database. Η αλλαγή γίνεται μόνο στα πεδία που ο διαχειριστής θα αλλάξει την τιμή του και όχι σε όλα τα πεδία.

Στην κορυφή της διεπαφής βρίσκεται ένα Relative Layout που περιέχει ένα Text View με την επωνυμία του σταθμού και ένα Image Button στα δεξιά.

Πατώντας το Image Button ο διαχειριστής μεταφέρεται στις κρατήσεις θέσεων στάθμευσης που έχουν γίνει από χρήστες στον σταθμό του. Η διεπαφή αυτή φαίνεται σε portrait και landscape στις Εικόνες 41 και 42 αντίστοιχα.



Εικόνα 41: Οι κρατήσεις για τον σταθμό portrait



Εικόνα 42: Οι κρατήσεις για τον σταθμό landscape

Η διεπαφή αυτή στην κορυφή έχει ένα Relative Layout που περιέχει ένα Image Button όπου πατώντας το ο διαχειριστής επιστρέφει στην διεπαφή προφιλ του σταθμού και ένα Text View με το κείμενο Κρατήσεις. Οι κρατήσεις φαίνονται σε μορφή card View με την χρήση recycler view. Το recycler view βρίσκεται μέσα σε ένα δεύτερο Relative Layout μαζί με ένα Scroll View.

2.8 Επίλογος

Στο κεφάλαιο παρουσιάστηκαν όλες οι επιμέρους διεπαφές της εφαρμογής με σκοπό την κατανόηση των λειτουργιών της παρούσας εφαρμογής με σκοπό την ευκολότερη κατανόηση του κώδικα της εφαρμογής που βρίσκεται πίσω από κάθε Activity και των επιμέρους κλάσεων που θα παρουσιαστούν στο επόμενο κεφάλαιο.

Κεφάλαιο 3ο: Ανάλυση Λειτουργιών της Εφαρμογής

3.1 Εισαγωγή

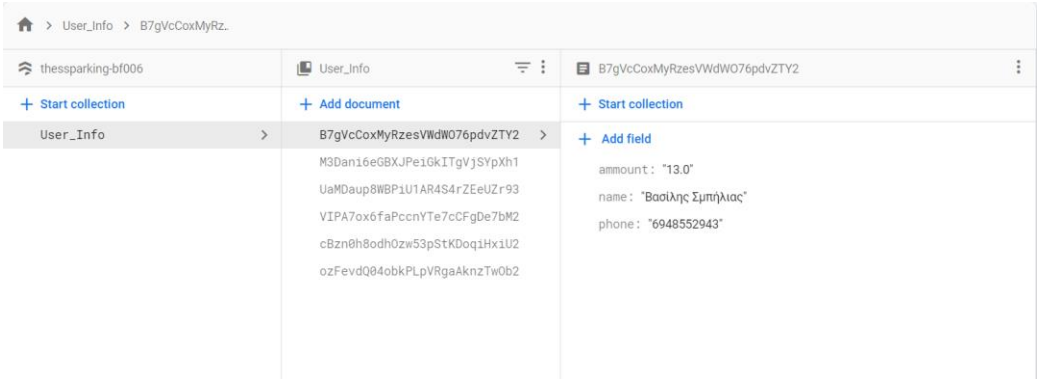
Σε αυτό το κεφάλαιο θα αναλυθούν τα επιμέρους συστατικά της εφαρμογής και θα παρουσιαστεί ο κώδικας της.

3.2 Βάση Δεδομένων

Για την αποθήκευση των δεδομένων όπως αναφέραμε και προηγουμένων χρησιμοποιήθηκε η Firebase της Google. Στην Firestore Database έχουμε μία συλλογή η οποία αποθηκεύει τα στοιχεία του χρήστη, ενώ στην Realtime Database έχουμε τρεις συλλογές μία για την αποθήκευση των σταθμών στάθμευσης, μία για την αποθήκευση των αγαπημένων σταθμών του χρήστη και μία για την αποθήκευση των κρατήσεων του χρήστη.

3.2.1 Στοιχεία χρήστη

Στην συλλογή User_Info υπάρχουν documents όπου το κάθε document αποτελεί και τα στοιχεία ενός χρήστη που είναι εγγεγραμμένος στην εφαρμογή. Ο τίτλος κάθε document είναι ίδιος με το UserId του χρήστη. Η συλλογή με τα documents και τα πεδία του κάθε document φαίνεται στην Εικόνα 43.



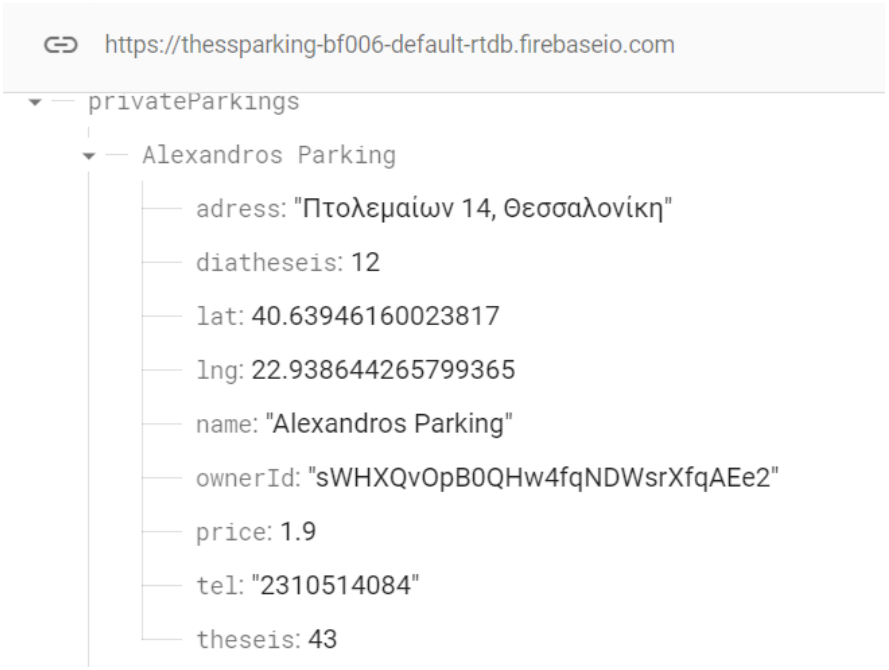
Εικόνα 43: Συλλογή User_Info

Κάθε collection αποτελείται από τρία πεδία τα οποία είναι τα εξής:

- amount. Είναι τύπου String και περιέχει το διαθέσιμο υπόλοιπο του χρήστη.
- name. Είναι τύπου String και περιέχει το όνομα του χρήστη.
- phono. Είναι τύπου String και περιέχει τηλέφωνο του χρήστη.

3.2.2 Σταθμοί στάθμευσης

Για αυτήν την συλλογή και για τις επόμενες δύο χρησιμοποιήθηκε η Realtime Database. Μέσα στην συλλογή privateParkings υπάρχουν εγγραφές όπου η κάθε εγγραφή αποτελεί έναν σταθμό στάθμευσης. Η κάθε εγγραφή έχει για τίτλο την επωνυμία του αντίστοιχου σταθμού στάθμευσης όπως φαίνεται στην Εικόνα 44.



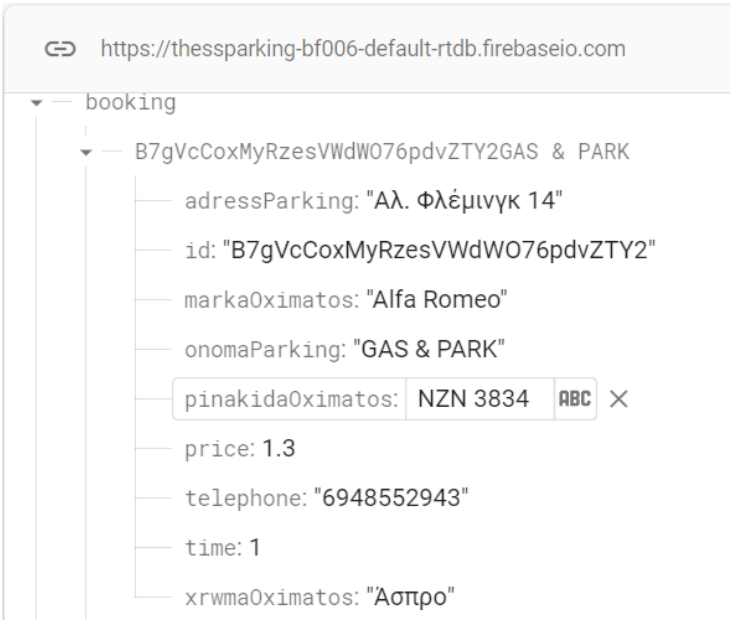
Εικόνα 44: Η συλλογή privateParking με μία εγγραφή της.

Κάθε εγγραφή στην συλλογή αυτή αποτελείται από τα εξής πεδία:

- address. Είναι τύπου String και περιέχει την διεύθυνση του σταθμού στάθμευσης.
- diatheseis. Είναι τύπου int και περιέχει τον διαθέσιμο αριθμό θέσεων του σταθμού στάθμευσης.
- lat. Είναι τύπου double και περιέχει το γεωγραφικό πλάτος του σταθμού στάθμευσης.
- lng. Είναι τύπου double και περιέχει το γεωγραφικό μήκος του σταθμού στάθμευσης.
- name. Είναι τύπου String και περιέχει την επωνυμία του σταθμού στάθμευσης.
- ownerId. Είναι τύπου String και περιέχει το id του διαχειριστή του σταθμού στάθμευσης.
- price. Είναι τύπου double και περιέχει την τιμή χρέωσης ανά ώρα του σταθμού στάθμευσης.
- tel. Είναι τύπου String και περιέχει το τηλέφωνο επικοινωνίας του σταθμού στάθμευσης.
- theseis. Είναι τύπου int και περιέχει τον συνολικό αριθμό θέσεων που διαθέτει ο σταθμού στάθμευσης.

3.2.3 Κρατήσεις χρήστη

Οι κρατήσεις του χρήστη περιέχονται στην συλλογή booking. Η συλλογή αυτή περίεχει μία εγγραφή για κάθε κράτηση η οποία φέρει τον τίτλο του id του χρήστη που κάνει την κράτηση σε συνδυασμό με το όνομα του σταθμού στάθμευσης όπως φαίνεται στην Εικόνα 45.



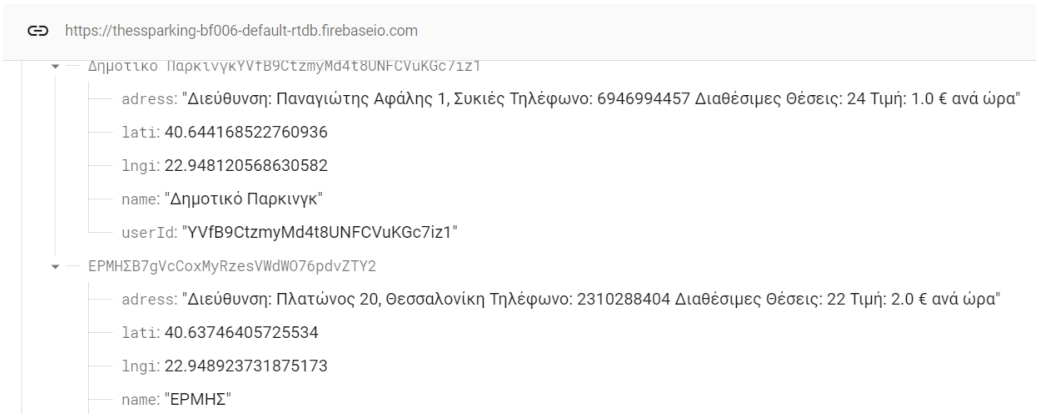
Εικόνα 45: Η συλλογή booking με μία εγγραφή της

Κάθε εγγραφή αποτελείται από τα εξής πεδία:

- adressParking. Είναι τύπου String και περιέχει την διεύθυνση του σταθμού στάθμευσης.
- id. Είναι τύπου String και περιέχει το id του χρήστη που έκανε την κράτηση.
- markaOximatos. Είναι τύπου String και περιέχει την μάρκα του οχήματος του χρήστη.
- onomaParking. Είναι τύπου String και περιέχει το όνομα του σταθμού στάθμευσης.
- pinakidaOximatos. Είναι τύπου String και περιέχει τον Αριθμό κυκλοφορίας του οχήματος του χρήστη.
- price. Είναι τύπου double και περιέχει την συνολική χρέωση του χρήστη για την κράτηση θέσης στον σταθμό στάθμευσης.
- telephone. Είναι τύπου String και περιέχει το τηλέφωνο του χρήστη.
- time. Είναι τύπου int και περιέχει τον χρόνο που επέλεξε ο χρήστης για την στάθμευση του οχήματος του στον σταθμό.
- xrwmaOximatos. Είναι τύπου String και περιέχει το χρώμα του οχήματος του χρήστη.

3.2.4 Αγαπημένα χρήστη

Τα αγαπημένα του χρήστη αποθηκεύονται στην συλλογή parkings η οποία περιέχει εγγραφές που έχουν ως τίτλο το όνομα του σταθμού σε συνδυασμό με το id του χρήστη όπως φαίνεται στην Εικόνα 46:



Εικόνα 46: Εγγραφές της συλλογής parkings

Κάθε εγγραφή αποτελείται από τα παρακάτω εξής πεδία.

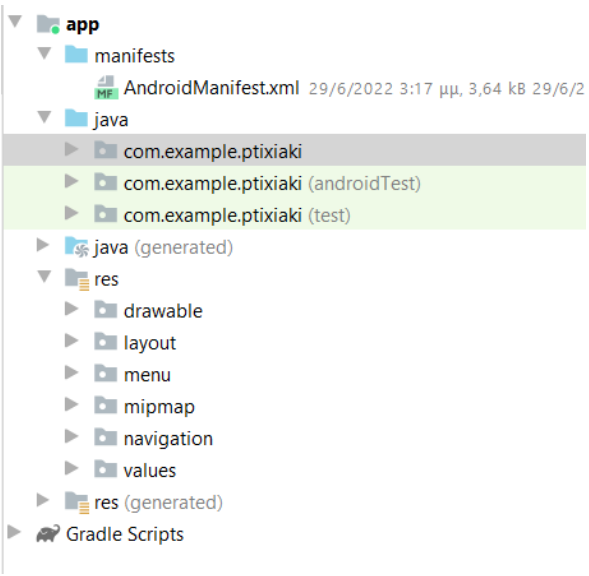
- address. Είναι τύπου String και περιέχει την διεύθυνση του σταθμού στάθμευσης, καθώς και πληροφορίες σχετικά με αυτόν όπως το τηλέφωνο, τις διαθέσιμες θέσεις του και την τιμή του ανά ώρα.
- lati. Είναι τύπου double και περιέχει το γεωγραφικό πλάτος του σταθμού στάθμευσης.
- lngi. Είναι τύπου double και περιέχει το γεωγραφικό μήκος του σταθμού στάθμευσης.
- name. Είναι τύπου String και περιέχει το όνομα του σταθμού στάθμευσης.
- userId. Είναι τύπου String και περιέχει το id του χρήστη.

3.3 Δομή του project στο Android Studio

Στην Εικόνα 47 φαίνεται η δομή ενός project στο Android Studio. Στην αρχή υπάρχει το αρχείο AndroidManifest.xml το οποίο περιγράφει την εφαρμογή.

Ακριβώς από κάτω στον φάκελο java υπάρχουν τρεις υποφάκελοι. Στον πρώτο υποφάκελο περιέχεται όλος ο κώδικας της εφαρμογής, δηλαδή τα activities και οι κλάσεις. Ο δεύτερος και ο τρίτος υποφάκελος περιέχουν αρχεία java που χρησιμοποιούνται για τον έλεγχο της σωστής λειτουργίας της εφαρμογής.

Στον φάκελο res υπάρχουν αρχεία τα οποία χρησιμοποιούνται για την σχεδίαση των διεπαφών της εφαρμογής. Τα αρχεία αυτά χωρίζονται ανα είδος σε υποφακέλους. Στον φάκελο drawable υπάρχουν όλα τα αρχεία εικόνων που χρησιμοποιούνται στην εφαρμογή. Στον φάκελο layout υπάρχουν όλες οι διεπαφές της εφαρμογής σε xml, δηλαδή για κάθε activity υπάρχει και ένα αντίστοιχο xml αρχείο. Στον φάκελο menu έχουμε τα δύο μενού που χρησιμοποιούνται στην εφαρμογή, δηλαδή το bottom navigation και το popup menou. Στον φάκελο mipmap υπάρχει το εικονίδιο που χρησιμοποιείται για την εφαρμογή σε πέντε διαφορετικά μεγέθη. Τέλος στον φάκελο values περιλαμβάνονται αρχεία σχετικά με τα χρώματα που χρησιμοποιούνται στην εφαρμογή όπως το colors.xml, το style που χρησιμοποιείται, διάφορα strings κ.α.



Εικόνα 47: Δομή του project στο Android Studio

3.4 AndroidManifest.xml

Το AndroidManifest.xml είναι απαραίτητο αρχείο για κάθε Android project καθώς περιλαμβάνει όλες τις απαραίτητες πληροφορίες που χρειάζονται για την δημιουργία του project. Στο AndroidManifest.xml δηλώνεται το όνομα της εφαρμογής, τα activities από τα οποία αποτελείται

(components), τα receivers, τα services, καθώς και τα δικαιώματα που απαιτούνται για την λειτουργία της εφαρμογής.

Το αρχείο AndroidManifest.xml φαίνεται στις Εικόνες 48 και 49

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    package="com.example.ptixiaki">

    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
    <uses-permission android:name="com.google.android.providers.gsf.permission.READ_GSERVICES" />
    <uses-permission
        android:name="com.google.android.gms.permission.AD_ID"
        tools:node="remove" />

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_parking_2"
        android:label="Thess Parking"
        android:roundIcon="@mipmap/ic_parking_profile2_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.Ptixiaki"
        android:usesCleartextTraffic="true">
        <activity android:name=".KratiseisActivity"></activity>
        <activity android:name=".ParkingOwnerActivity" />
        <activity android:name=".ParkingOwnerSignUpActivity" />
        <activity android:name=".ParkingOwnerSignInActivity" />
        <activity android:name=".BookingActivity" />

        <meta-data
            android:name="com.google.android.geo.API_KEY"
            android:value="AIzaSyCdRr-4fM6xO9xj8UEkIVGcqNSEgS11kUE" />
```

Εικόνα 48: AndroidManifest.xml(α)

```
        <activity
            android:name=".MapsActivity"
            android:label="Map" />
        <activity android:name=".ChangeEmailActivity" />
        <activity android:name=".ChangePasswordActivity" />
        <activity android:name=".ProfileActivity" />
        <activity android:name=".FavoritesActivity" />
        <activity android:name=".SignUpActivity" />
        <activity android:name=".LoginActivity" />
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
```

Εικόνα 49: AndroidManifest.xml(β)

Όταν μια εφαρμογή γίνεται build την θέση του package name παίρνει το label στο application. Επίσης στο AndroidManifest.xml δηλώνονται και τα δικαιώματα που χρησιμοποιούνται για την λειτουργία της εφαρμογής όπως φαίνεται στην Εικόνα 48 στα label “uses-permission”. Επίσης δηλώνονται όλα τα

activities που αποτελούν την εφαρμογή καθώς και το API key που χρησιμοποιείται για την χρήση των τεχνολογιών της Google από την εφαρμογή.

3.5 Δικαιώματα

Τα δικαιώματα σε μία εφαρμογή είναι απαραίτητα για να έχει πρόσβαση σε διάφορα προστατευμένα μέρη του συστήματος ή άλλων εφαρμογών. Αντίστοιχα για να έχουν πρόσβαση άλλες εφαρμογές στην συγκεκριμένη εφαρμογή πρέπει αυτές να δηλώσουν το αντίστοιχο δικαίωμα για την εφαρμογή αυτή.

Όπως φαίνεται και στην Εικόνα 48 τις προηγούμενης ενότητας η δήλωση των δικαιωμάτων στο AndroidManifest.xml γίνεται με την χρήση της εντολής uses-permission χρησιμοποιώντας το label android:name όπου μέσα σε αυτό δηλώνεται το δικαίωμα.

Δικαιώματα που χρησιμοποιήθηκαν στην συγκεκριμένη εφαρμογή είναι την ανάκτηση της τοποθεσίας του χρήστη και την πρόσβαση της εφαρμογής στο διαδίκτυο και παροχή του αριθμού των συσκευών που χρησιμοποιούν τις υπηρεσίες της google κάνοντας χρήση της εφαρμογής.

3.6 Αρχεία Gradle

Τα αρχεία Gradle δημιουργούνται κατά την δημιουργία του project στο Android Studio και είναι ένα σύνολο εργαλείων build που χρησιμοποιείται για την αυτοματοποίηση και την διαχείριση της διαδικασίας build. Υπάρχουν αρκετά αρχεία Gradle αλλά εμείς θα εστιάσουμε στα δύο αρχεία build.gradle.

3.6.1 Top-level build file build.gradle (Ptixiaki)

Σε αυτό το αρχείο καθορίζονται οι ρυθμίσεις του build που ισχύουν σε όλα τα modules, τα repositories και τα dependencies που είναι κοινά στο project. Το αρχείο αυτό φαίνεται στην Εικόνα 50.

```
// Top-level build file where you can add configuration options common to all sub-projects/modules.
buildscript {
    ext {
        kotlin_version = '1.3.72'
    }
    repositories {
        google()
        jcenter()
    }
    dependencies {
        classpath "com.android.tools.build:gradle:4.1.3"
        classpath 'com.google.gms:google-services:4.3.10'
        classpath "org.jetbrains.kotlin:kotlin-gradle-plugin:$kotlin_version"
        // NOTE: Do not place your application dependencies here; they belong
        // in the individual module build.gradle files
    }
}

allprojects {
    repositories {
        google()
        jcenter()
        mavenCentral()
    }
}

task clean(type: Delete) {
    delete rootProject.buildDir
}
```

Εικόνα 50: build.gradle (Ptixiaki)

3.6.2 Module level build file build.gradle (:app)

Το build.gradle(:app) αφορά το συγκεκριμένο Module και δηλώνονται βασικές πληροφορίες όπως η έκδοση του Android που τρέχει η εφαρμογή καθώς και τα dependencies που χρησιμοποιεί η εφαρμογή όπως η Google, το Google Maps, η Firebase κ.α.

Το αρχείο αυτό φαίνεται στην Εικόνα 51. Στην Εικόνα 51 φαίνονται μερικά από τα dependencies που χρησιμοποιεί η εφαρμογή.

```
plugins {
    id 'com.android.application'
    id 'com.google.gms.google-services'
    //id 'kotlin-android'
}

android {
    compileSdkVersion 30
    buildToolsVersion "30.0.3"

    defaultConfig {
        applicationId "com.example.ptixiaki"
        minSdkVersion 21
        targetSdkVersion 30
        versionCode 1
        versionName "1.0"

        testInstrumentationRunner "androidx.test.runner.AndroidJUnitRunner"
    }

    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'), 'proguard-rules.pro'
        }
    }
    compileOptions {
        sourceCompatibility JavaVersion.VERSION_1_8
        targetCompatibility JavaVersion.VERSION_1_8
    }
}

dependencies {
    implementation 'com.github.bumptech.glide:glide:4.12.0'
    // Glide v4 uses this new annotation processor -- see https://bumptech.github.io/glide/doc/generatedapi.html
    annotationProcessor 'com.github.bumptech.glide:compiler:4.12.0'
    //gia to PolyUtil
    implementation 'com.google.maps.android:android-maps-utils:0.5+'
    implementation 'androidx.annotation:annotation:1.2.0'
    implementation 'androidx.lifecycle:lifecycle-livedata-ktx:2.3.1'
    implementation 'androidx.lifecycle:lifecycle-viewmodel-ktx:2.3.1'
    implementation 'com.google.android.material:material:1.3.0'
    implementation 'androidx.legacy:legacy-support-v4:1.0.0'
    implementation 'com.google.firebase:firebase-analytics:19.0.0'
    implementation 'com.google.firebase:firebase-auth:21.0.1'
    implementation 'com.google.android.gms:play-services-maps:17.0.1'
    implementation "org.jetbrains.kotlin:kotlin-stdlib:$kotlin_version"
    implementation 'com.google.firebase:firebase-database:20.0.5'
    implementation 'com.google.firebase:firebase-firestore:24.1.2'
    def nav_version = "2.3.5"
    // Java language implementation
    implementation "androidx.navigation:navigation-fragment:$nav_version"
    implementation "androidx.navigation:navigation-ui:$nav_version"
    // Feature module Support
    implementation "androidx.navigation:navigation-dynamic-features-fragment:$nav_version"
    // Testing Navigation
    androidTestImplementation "androidx.navigation:navigation-testing:$nav_version"
    // Jetpack Compose Integration
    implementation "androidx.navigation:navigation-compose:2.4.0-alpha02"
    implementation 'androidx.appcompat:appcompat:1.3.0'
    implementation 'com.google.android.material:material:1.3.0'
    implementation 'androidx.constraintlayout:constraintlayout:2.0.4'
```

Εικόνα 51: build.gradle (:app)

3.7 Κλάσεις

Οι κλάσεις που χρησιμοποιούνται χωρίζονται στις κλάσεις που μας βοηθούν στην αναπαράσταση δεδομένων και στην αποθήκευση τους στην βάση δεδομένων και τις κλάσεις για την αναπαράσταση των αγαπημένων του χρήστη και των κρατήσεων στο κομμάτι του διαχειριστή στα δύο recycler view αντίστοιχα.

Οι κλάσεις για την αναπαράσταση δεδομένων είναι οι Booking.java, Parkings.java και UserFav.java όπως φαίνεται στις Εικόνες 52, 53 και 54.

```java
package com.example.ptixiaki;

public class Booking {
    String id;
    String pinakidaOximatos;
    String xrwmaOximatos;
    String markaOximatos;
    String telephone;
    String onomaParking;
    String adressParking;
    double price;
    int time;
```

Εικόνα 52: Booking.java

```java
package com.example.ptixiaki;

public class Parkings {
    private String name;
    private String adress;
    private String tel;
    private int theseis;
    private int diatheseis;
    private double price;
    private double lat;
    private double lng;
    private String ownerId;
```

Εικόνα 53: Parkings.java

```java
package com.example.ptixiaki;

public class UserFav {
    private String name;
    private String adress;
    private String tel;
    private double lati;
    private double lngi;
    private String userId;
```

Εικόνα 54: UserFav.java

Η κάθε κλάση περιέχει τα πεδία της, δύο δομητές έναν κενό και έναν με όλα τα πεδία και τις μεθόδους getter.

Οι κλάσεις για την αναπαράσταση των αγαπημένων και των κρατήσεων είναι οι MyAdapter.java και MyAdapterBooking.java. Οι δύο αυτές κλάσεις έχουν την ίδια λειτουργία οπότε θα αναλυθεί η πρώτη κλάση MyAdapter.java.

```java
package com.example.ptixiaki;

import ...

public class MyAdapter extends RecyclerView.Adapter<MyAdapter.MyViewHolder> {
    Context context;
    ArrayList<UserFav> list;

    RecyclerView recyclerView;
    DatabaseReference databaseReference;

    public MyAdapter(Context context, ArrayList<UserFav> list) {
        this.context = context;
        this.list = list;
    }

    @NonNull
    @Override
    public MyViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
        View v = LayoutInflater.from(context).inflate(R.layout.userfav, parent, attachToRoot: false);
        return new MyViewHolder(v);
    }
```

Εικόνα 55: MyAdapter.java(α)

Στην κλάση MyAdapter.java δηλώνεται το context, ένα ArrayList τύπου UserFav, το recyclerview και το databaseReference που μας βοηθά να έχουμε πρόσβαση στην βάση δεδομένων. Στην κλάση υπάρχει και ένας δομητής. Η μέθοδος onCreateViewHolder δημιουργεί ένα layout τύπου userfav και το επιστρέφει στον χρήστη. Το layout userfav δημιουργήθηκε σε xml και αποτελείται από ένα card view που μέσα σε αυτό εμφανίζονται κάποιες πληροφορίες του σταθμού στάθμευσης.

Στην Εικόνα 65 φαίνεται η μέθοδος onBindViewHolder στην οποία ελέγχεται αν το uid του χρήστη που είναι συνδεδεμένος είναι ίδιο με το πεδίο userId του σταθμού στάθμευσης. Αυτός ο έλεγχος γίνεται για να εμφανίζεται στον χρήστη μόνο τα δικά του αγαπημένα και όχι όλα τα αγαπημένα όλων των χρηστών που υπάρχουν στην βάση δεδομένων. Εάν τα δύο πεδία είναι ίδια η κάρτα με τις πληροφορίες του σταθμού εμφανίζεται στα αγαπημένα του χρήστη, ενώ εάν όχι η κάρτα ‘εξαφανίζεται’.

```java
    @Override
    public void onBindViewHolder(@NonNull MyViewHolder holder, int position) {
        UserFav user = list.get(position);
        String uid = user.getUserId();
        if ((uid.equals(FirebaseAuth.getInstance().getCurrentUser().getUid()))) {
            holder.textnamepar.setText(user.getName());//edw pername ta dedomena
            holder.textdieupar.setText("" + user.getAdress());//edw pername ta dedomena
            holder.textlati.setText(" " + user.getLati());//edw pername ta dedomena
            holder.textlngi.setText(" " + user.getLngi());//edw pername ta dedomena
        }
        //Na min emfanizontai ta agapimena twn allwn xristwn
        else if (!(uid.equals(FirebaseAuth.getInstance().getCurrentUser().getUid()))) {
            holder.linear.setVisibility(View.GONE);
            holder.cardFav.setVisibility(View.GONE);
            holder.linear7.setVisibility(View.GONE);
        }
    }

    @Override
    public int getItemCount() { return list.size(); }
```

Εικόνα 56: MyAdapter.java(β)

Στην Εικόνα 66 όπως φαίνεται δημιουργήθηκε η κλάση MyViewHolder μέσα στην MyAdapter στην οποία δηλώνονται όλα τα πεδία από το userfav.xml και εμφανίζονται στα αγαπημένα.

```
public class MyViewHolder extends RecyclerView.ViewHolder implements View.OnClickListener, PopupMenu.OnMenuItemClickListener {
    TextView textnamepar, textdieupar, textlati, textlngi;
    ImageButton imageButton;
    MenuItem nav, del;
    LinearLayout linear, linear7;
    androidx.cardview.widget.CardView cardFav;
    public MyViewHolder(@NonNull View itemView) {
        super(itemView);
        textnamepar = itemView.findViewById(R.id.textnamepar);
        textdieupar = itemView.findViewById(R.id.textdieupar);
        textlati = itemView.findViewById(R.id.textlati);
        textlngi = itemView.findViewById(R.id.textlngi);
        imageButton = itemView.findViewById(R.id.imageButton);
        cardFav = (androidx.cardview.widget.CardView) itemView.findViewById(R.id.cardFav);
        imageButton.setOnClickListener(this);
        linear = (LinearLayout)itemView.findViewById(R.id.linear);
        linear7 = (LinearLayout)itemView.findViewById(R.id.linear7);
        nav = itemView.findViewById(R.id.action_popup_nav);
        del = itemView.findViewById(R.id.action_popup_delete);
        recyclerView = itemView.findViewById(R.id.recycleview);
    }
```

Εικόνα 57: MyAdapter.java(γ)

```
@Override
public boolean onMenuItemClick(MenuItem item) {
    switch (item.getItemId()) {
        case R.id.action_popup_nav:
            Uri gmmIntentUri = Uri.parse("google.navigation:q="+textlati.getText().toString()+","+textlngi.getText().toString());
            Intent mapIntent = new Intent(Intent.ACTION_VIEW, gmmIntentUri);
            mapIntent.setPackage("com.google.android.apps.maps");
            MyAdapter.this.context.startActivity(mapIntent);
            return true;
        case R.id.action_popup_book:
            Intent intentBook = new Intent(getApplicationContext(), BookingActivity.class);
            intentBook.putExtra( name: "parking_name", textnamepar.getText().toString());
            MyAdapter.this.context.startActivity(intentBook);
            return true;
        case R.id.action_popup_delete:
            String name = textnamepar.getText().toString()+""+ FirebaseAuth.getInstance().getCurrentUser().getUid();
            deleteParking(name);
            Intent intent = new Intent(getApplicationContext(), FavoritesActivity.class);
            MyAdapter.this.context.startActivity(intent);
            return true;
        default:
            return false;
    }
}
```

Εικόνα 58: MyAdapter.java(δ)

Στην Εικόνα 58 φαίνεται η μέθοδος onMenuItemClick. Όπως έχει αναφερθεί προηγουμένος σε κάθε card view που βλέπει ο χρήστης σαν αγαπημένο του υπάρχει ένα popup menu το οποίο περιέχει 3 επιλογές οι οποίες είναι η πλοήγηση προς τον αγαπημένο σταθμό, η κράτηση θέσης σε αυτόν και η διαγραφή του από τα αγαπημένα. Στην περίπτωση της πλοήγησης δημιουργούμε ένα URI στο οποίο περνάμε το γεωγραφικό πλάτος και μήκος του σταθμού και εκκινούμε το gps του google δίνοντας ως αφετερία την τοποθεσία μας και προορισμό τις συντεταγμένες του σταθμού. Στην περίπτωση της κράτησης εκκινούμε το Booking Activity και περνάμε σε αυτό ως έξτρα πληροφορία το όνομα του σταθμού. Τέλος στην περίπτωση της διαγραφής του σταθμού από τα αγαπημένα καλούμε την μέθοδο deleteParking, περνάμε ως παράμετρο το όνομα του σταθμού και το uid του χρήστη τα οποία αποτελούν τον τίτλο της κάθε εγγραφής των αγαπημένων σταθμών. Αφού καλέσουμε την μέθοδο deleteParking κάνουμε refresh το Activity των αγαπημένων. Στην μέθοδο deleteParking χρησιμοποιείται η μέθοδος της firebase removeValue για την διαγραφή της συγκεκριμένης εγγραφής από την βάση δεδομένων.

3.8 Activities

Στην παρούσα ενότητα θα αναλυθούν όλα τα activities που απαρτίζουν την εφαρμογή.

3.8.1 Main Activity

Το Main Activity είναι αυτό που τρέχει όταν ξεκινάει η εφαρμογή καθώς είναι δηλωμένο στο αρχείο AndroidManifest.xml. Στην μέθοδο onCreate όπως φαίνεται και στην Εικόνα 59 γίνεται έλεγχος εάν υπάρχει χρήστης συνδεδεμένος στην εφαρμογή. Εάν υπάρχει τότε ή πρώτη εικόνα που θα δει ο χρήστης είναι το Maps Activity, ενώ εάν δεν υπάρχει συνδεδεμένος χρήστης τότε στον χρήστη θα εμφανιστεί το Login Activity.

```
1  package com.example.ptixiaki;
2
3  import ...
11
12 public class MainActivity extends AppCompatActivity {
13     private Handler handler;
14     private FirebaseAuth auth;
15     private FirebaseUser user;
16
17     @Override
18     protected void onCreate(Bundle savedInstanceState) {
19         super.onCreate(savedInstanceState);
20         //setContentView(R.layout.activity_main);
21         auth = FirebaseAuth.getInstance();
22         if (auth.getCurrentUser() != null) {
23             startActivity(new Intent(getApplicationContext(), MapsActivity.class));
24         }
25         else {
26             startActivity(new Intent(getApplicationContext(), LoginActivity.class));
27         }
28     }
29 }
```

Εικόνα 59: Main Activity

3.8.2 Login Activity & ParkingOwnerSignIn Activity

Εφόσον ο χρήστης δεν είναι συνδεδεμένος μεταβαίνει στο Login Activity. Το Login Activity όπως παρουσιάστηκε και στο προηγούμενο κεφάλαιο είναι παρόμοιο με το ParkingOwnerSignIn Activity με τη μόνη διαφορά τους να είναι η δυνατότητα σύνδεσης του χρήστη στο Login Activity με επιπλέον επιλογές όπως η σύνδεση μέσω google και github, ενώ στην σύνδεση του διαχειριστή κάποιου σταθμού στάθμευσης υπάρχει η δυνατότητα σύνδεσης μόνο με email και password.

Για την σύνδεση του χρήστη και του διαχειριστή στην εφαρμογή ο τρόπος είναι ακριβώς ίδιος. Εφόσον γίνει έλεγχος αν τα πεδία του email και του κωδικού πρόσβασης είναι συμπληρωμένα και ο κωδικός είναι μεγαλύτερος από 6 χαρακτήρες όπως φαίνεται στην Εικόνα 60, καλείται η μέθοδος signInWithEmailAndPassword και εφόσον η διαδικασία είναι επιτυχής ο χρήστης μεταβαίνει στο Maps Activity.

```
181         loginBtn.setOnClickListener(new View.OnClickListener() {
182             @Override
183             public void onClick(View v) {
184                 String email = editTextEmail.getText().toString().trim();
185                 String password = editTextPassword.getText().toString().trim();
186
187                 if (TextUtils.isEmpty(email)) {
188                     editTextEmail.setError("Email is required");
189                     return;
190                 }
191
192                 if (TextUtils.isEmpty(password)) {
193                     editTextPassword.setError("Password is required");
194                     return;
195                 }
196
197                 if (password.length() < 6) {
198                     editTextPassword.setError("Password must be at least 6 characters");
199                     return;
200                 }
201
202                 //Authentication of User
203                 auth.signInWithEmailAndPassword(email, password).addOnCompleteListener(new OnCompleteListener<AuthResult>() {
204                     @Override
205                     public void onComplete(@NonNull Task<AuthResult> task) {
206                         if (task.isSuccessful()) {
207                             progressBar.setVisibility(View.VISIBLE);
208                             startActivity(new Intent(getApplicationContext(), MapsActivity.class));
209                         }
210                         else {
211                             Toast.makeText( context: LoginActivity.this,  text: "Error... " + task.getException().getMessage(), Toast.LENGTH_SHORT).show();
212                         }
213                     }
214                 });
215             }
```

Εικόνα 60: Σύνδεση του χρήστη με email και password

Για την επαναφορά του λογαριασμού εφόσον ο χρήστης έχει ξεχάσει τον κωδικό του γίνεται πατώντας το αντίστοιχο text view. Όπως φαίνεται και στην παρακάτω Εικόνα, πατώντας το δημιουργείται ένα Alert Dialog στο οποίο ο χρήστης συμπληρώνει το email του.

```
220         textViewForget.setOnClickListener(new View.OnClickListener() {
221             @Override
222             public void onClick(View v) {
223                 EditText resetEmail = new EditText(v.getContext());
224                 AlertDialog.Builder passwordResetDialog = new AlertDialog.Builder(v.getContext());
225                 passwordResetDialog.setTitle("Reset Password?");
226                 passwordResetDialog.setMessage("Enter your Email to reset received link.");
227                 passwordResetDialog.setView(resetEmail);
228
229                 passwordResetDialog.setPositiveButton( text: "Yes", new DialogInterface.OnClickListener() {
230                     @Override
231                     public void onClick(DialogInterface dialog, int which) {
232                         String mail = resetEmail.getText().toString();
233                         auth.sendPasswordResetEmail(mail).addOnSuccessListener(new OnSuccessListener<Void>() {
234                             @Override
235                             public void onSuccess(Void aVoid) {
236                                 Toast.makeText( context: LoginActivity.this,  text: "Reset Link Sent to your Email", Toast.LENGTH_SHORT).show();
237                             }
238                         }).addOnFailureListener(new OnFailureListener() {
239                             @Override
240                             public void onFailure(@NonNull Exception e) {
241                                 Toast.makeText( context: LoginActivity.this,  text: "Error! Reset Link is Not Sent" + e.getMessage(), Toast.LENGTH_SHORT).show()
242                             }
243                         });
244                     }
245                 });
246
247                 passwordResetDialog.setNegativeButton( text: "No", new DialogInterface.OnClickListener() {
248                     @Override
249                     public void onClick(DialogInterface dialog, int which) {
250                         //close the dialog
251                     }
252                 });
253                 passwordResetDialog.create().show();
254             }
```

Εικόνα 61: Επαναφορά κωδικού πρόσβασης

Συμπληρώνοντας το email του ο χρήστης και πατώντας το Yes καλείται η μέθοδος sendPasswortResetEmail και ο χρήστης λαμβάνει email επαναφοράς κωδικού και συμπληρώνοντας τον καινούργιο του κωδικό γίνεται reauthentication του λογαριασμού του στην firebase.

Για την σύνδεση του χρήστη μέσω github ο χρήστης πρέπει να συμπληρώσει το email που χρησιμοποιεί στον λογαριασμό github στο Alert Dialog που εμφανίζεται όταν πατηθεί το image view του github.

Στην Εικόνα 62 φαίνεται ο τρόπος υλοποίησης της σύνδεσης στην εφαρμογή μέσω github.

```
githubImageView.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        EditText gitEmail = new EditText(v.getContext());
        AlertDialog.Builder githubDialog = new AlertDialog.Builder(v.getContext());
        githubDialog.setTitle("Το email σου ");
        githubDialog.setMessage("Το email του github λογαριασμού σου ");
        githubDialog.setView(gitEmail);

        githubDialog.setPositiveButton( text: "Ok", new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
                String email = gitEmail.getText().toString();
                OAuthProvider.Builder provider = OAuthProvider.newBuilder( providerId: "github.com");
                provider.addCustomParameter( paramKey: "login", email);
                List<String> scopes =
                        new ArrayList<String>() {
                            {
                                add("user:email");
                            }
                        };
                provider.setScopes(scopes);
                Task<AuthResult> pendingResultTask = auth.getPendingAuthResult();
                if (pendingResultTask != null) {
                    pendingResultTask.addOnSuccessListener(new OnSuccessListener<AuthResult>() {
                        @Override
                        public void onSuccess(AuthResult authResult) {


                        }
                    }).addOnFailureListener(new OnFailureListener() {
                        @Override
                        public void onFailure(@NonNull Exception e) {
                            Toast.makeText( context: LoginActivity.this,  text: "Error... " + e.getMessage(), Toast.LENGTH_SHORT).show();
                        }
                    });
                } else {
                    auth.startActivityForSignInWithProvider( activity: LoginActivity.this, provider.build()).addOnSuccessListener(new OnSuccessListener
                        @Override
                        public void onSuccess(AuthResult authResult) {
                            progressBar.setVisibility(View.VISIBLE);
                            startActivity(new Intent( packageContext: LoginActivity.this, MapsActivity.class));
                        }
                    }).addOnFailureListener(new OnFailureListener() {
                        @Override
                        public void onFailure(@NonNull Exception e) {
                            Toast.makeText( context: LoginActivity.this,  text: "Error... " + e.getMessage(), Toast.LENGTH_SHORT).show();
                        }
                    });
                }
            }
        });
        githubDialog.create().show();
    }
});
```

Εικόνα 62: Υλοποίηση σύνδεσης μέσω Github

Για την υλοποίηση της σύνδεσης πρέπει εκτός από τον κώδικα ο οποίος είναι ο ίδιος που προτείνεται στην σελίδα της Firebase, πρέπει στο Sign In method του authentication στην firebase να ενεργοποιηθεί η επιλογή του github. Επίσης μέσω των ρυθμίσεων για Developers στο github πρέπει να δημιουργηθεί ένα OAuth Application και να συμπληρωθούν τα απαραίτητα πεδία εκεί. Επίσης πρέπει το Client Id που θα εμφανιστεί στο καινούργιο OAuth Application να συμπληρωθεί στο πεδίο API Key στην firebase. Στην συνέχεια κάνουμε generate ένα Client secret και το κάνουμε επικόλληση στο πεδίο API secret στην firebase.

Αντίστοιχα για την υλοποίηση της σύνδεσης μέσω google πρέπει να ενεργοποιηθεί το google sign In στο authentication της firebase και να υλοποιηθεί ο κώδικας ο οποίος παρέχεται μέσω του site της firebase και φαίνεται στις Εικόνες 63 και 64.

```
114         mAuth = FirebaseAuth.getInstance();
115         GoogleSignInOptions googleSignInOptions = new GoogleSignInOptions.Builder(GoogleSignInOptions.DEFAULT_SIGN_IN)
116                 .requestIdToken("18732249181-8gvsi99j8btn1n34k6gucfu92petfo9k.apps.googl...").requestEmail().build();
117         googleSignInClient = GoogleSignIn.getClient( activity: this, googleSignInOptions);
118         googleImageView.setOnClickListener(new View.OnClickListener() {
119             @Override
120             public void onClick(View v) { signIn(); }
121         });
```

Εικόνα 63: Υλοποίηση κώδικα για την σύνδεση μέσω google(α)

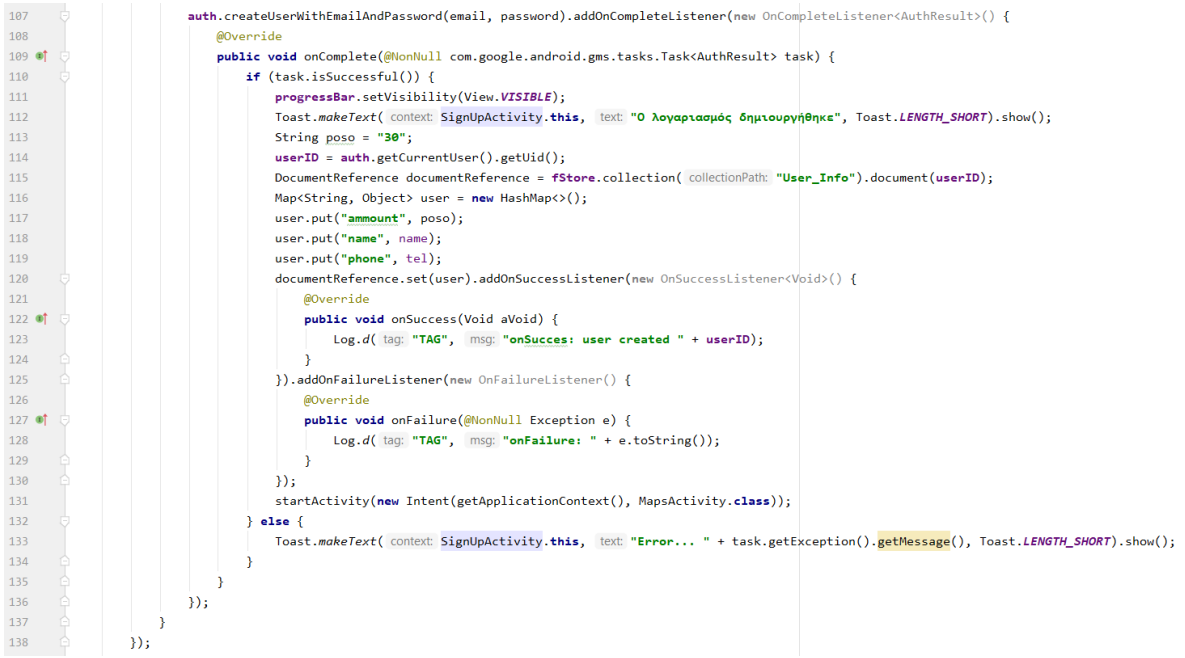
```
282     private void signIn() {
283         progressBar.setVisibility(View.VISIBLE);
284         Intent signInIntent = googleSignInClient.getSignInIntent();
285         startActivityForResult(signInIntent, RC_SIGN_IN);
286     }
287
288     @Override
289     protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent data) {
290         super.onActivityResult(requestCode, resultCode, data);
291         if (requestCode == RC_SIGN_IN) {
292             Task<GoogleSignInAccount> task = GoogleSignIn.getSignedInAccountFromIntent(data);
293             handleSignInResult(task);
294         }
295     }
296
297     private void handleSignInResult(Task<GoogleSignInAccount> completedTask) {
298         try {
299             GoogleSignInAccount acc =completedTask.getResult(ApiException.class);
300             //Toast.makeText(LoginActivity.this, "Signed In successfully", Toast.LENGTH_SHORT).show();
301             FirebaseGoogleAuth(acc);
302         }
303         catch (ApiException e) {
304             Toast.makeText( context: LoginActivity.this,  text: "SignIn Failed", Toast.LENGTH_SHORT).show();
305             FirebaseGoogleAuth( acct: null);
306         }
307     }
308
309     private void FirebaseGoogleAuth(GoogleSignInAccount acct) {
310         AuthCredential authCredential = GoogleAuthProvider.getCredential(acct.getIdToken(),  accessToken: null);
311         auth.signInWithCredential(authCredential).addOnCompleteListener( activity: this, new OnCompleteListener<AuthResult>() {
312             @Override
313             public void onComplete(@NonNull Task<AuthResult> task) {
314                 if (task.isSuccessful()) {
315                     FirebaseUser user = auth.getCurrentUser();
317                     startActivity(new Intent(getApplicationContext(), MapsActivity.class)); //edw prepei na se pairnei sto profile activity
318                 }
319                 else {
320                     Toast.makeText( context: LoginActivity.this,  text: "Failed", Toast.LENGTH_SHORT).show();
321                     updateUI( firebaseUser: null);
322                 }
323             }
324         });
325     }
```

Εικόνα 64: Υλοποίηση κώδικα για την σύνδεση μέσω google(β)

3.8.3 SignUp Activity & ParkingOwnerSignUp Activity

Το SignUp Activity και το ParkingOwnerSignUp Activity χρησιμοποιούνται για την εγγραφή του χρήστη στην εφαρμογή και την εγγραφή ενός σταθμού στάθμευσης από τον διαχειριστή του αντίστοιχα.

Τα δύο αυτά Activity υλοποιήθηκαν με παρόμοιο τρόπο. Για το SignUp Activity όπως φαίνεται και στην Εικόνα 65 αφού γίνει έλεγχος αφού γίνει αρχικοποίηση των μεταβλητών και έλεγχος εάν όλα τα πεδία είναι συμπληρωμένα, ο κωδικός πρόσβασης είναι τουλάχιστον έξι χαρακτήρες και το τηλέφωνο του χρήστη αποτελείται από δέκα αριθμούς γίνεται εγγραφή του χρήστη στην βάση δεδομένων με την χρήση της μεθόδου crateUserWithEmailAndPassword. Εφόσον η διαδικασία της εγγραφής είναι επιτυχής δημιουργείτε στο firestore ένα document που αφορά τις πληροφορίες του χρήστη και περίεχει τρία πεδία τα οποία είναι το όνομα του, το τηλέφωνο του και ένα αρχικό ποσό των τριάντα ευρώ, καθώς επίσης ο χρήστης μεταφέρεται στο Maps Activity.



```
            auth.createUserWithEmailAndPassword(email, password).addOnCompleteListener(new OnCompleteListener<AuthResult>() {
                @Override
                public void onComplete(@NonNull com.google.android.gms.tasks.Task<AuthResult> task) {
                    if (task.isSuccessful()) {
                        progressBar.setVisibility(View.VISIBLE);
                        Toast.makeText( context: SignUpActivity.this,  text: "Ο λογαριασμός δημιουργήθηκε", Toast.LENGTH_SHORT).show();
                        String poso = "30";
                        userID = auth.getCurrentUser().getUid();
                        DocumentReference documentReference = fStore.collection( collectionPath: "User_Info").document(userID);
                        Map<String, Object> user = new HashMap<>();
                        user.put("ammount", poso);
                        user.put("name", name);
                        user.put("phone", tel);
                        documentReference.set(user).addOnSuccessListener(new OnSuccessListener<Void>() {
                            @Override
                            public void onSuccess(Void aVoid) {
                                Log.d( tag: "TAG",  msg: "onSucces: user created " + userID);
                            }
                        }).addOnFailureListener(new OnFailureListener() {
                            @Override
                            public void onFailure(@NonNull Exception e) {
                                Log.d( tag: "TAG",  msg: "onFailure: " + e.toString());
                            }
                        });
                        startActivity(new Intent(getApplicationContext(), MapsActivity.class));
                    } else {
                        Toast.makeText( context: SignUpActivity.this,  text: "Error... " + task.getException().getMessage(), Toast.LENGTH_SHORT).show();
                    }
                }
            });
        }
    });
```

Εικόνα 65: Δημιουργεία λογαριασμού χρήστη

Αντίστοιχα για την εγγραφή ενός σταθμού στάθμευσης στην εφαρμογή από τον διαχειριστεί του αφού γίνει αρχικοποίηση των μεταβλητών και έλεγχος των πεδίων καλείται η ίδια μέθοδος όπως φαίνεται στην Εικόνα 66. Επίσης δημιουργείται ένα αντικείμενο τύπου Parkings και δημιουργείται μία εγγραφή στην Realtime Database με τίτλο το όνομα του σταθμού και στα πεδία της δίνονται οι τιμές που έχει συμπληρώσει ο διαχειριστής στα Edit Texts.

```
auth.createUserWithEmailAndPassword(email, password).addOnCompleteListener(new OnCompleteListener<AuthResult>() {
    @Override
    public void onComplete(@NonNull Task<AuthResult> task) {
        if (task.isSuccessful()) {
            simpleProgressBar2.setVisibility(View.VISIBLE);

            int seats = Integer.parseInt(synolikes_theseis);
            int available_seats = Integer.parseInt(diathesimes_theseis);
            double timi = Double.parseDouble(price);
            double lati = Double.parseDouble(lat);
            double lngi = Double.parseDouble(lng);
            String id = auth.getCurrentUser().getUid();
            Parkings parking = new Parkings(name, adress, phone, seats, available_seats, timi, lati, lngi, id);
            databaseReference.child("privateParkings").child(name).setValue(parking).addOnCompleteListener(new OnCompleteListener<Void>() {
                @Override
                public void onComplete(@NonNull Task<Void> task) {
                    if (task.isSuccessful()) {
                        Toast.makeText( context: ParkingOwnerSignUpActivity.this,  text: "Ο λογαριασμός Parking δημιουργήθηκε", Toast.LENGTH_SHORT).show();
                        startActivity(new Intent( packageContext: ParkingOwnerSignUpActivity.this, ParkingOwnerActivity.class));
                    }
                    else {
                        Toast.makeText(getApplicationContext(),  text: "Κατι πηγε στραβα", Toast.LENGTH_SHORT).show();
                    }
                }
            });

        }
        else {
            simpleProgressBar2.setVisibility(View.INVISIBLE);
            Toast.makeText( context: ParkingOwnerSignUpActivity.this,  text: "Error... " + task.getException().getMessage(), Toast.LENGTH_SHORT).show();
        }
    }
});
```

Εικόνα 66: Εγγραφή σταθμού στάθμευσης στην εφαρμογή

3.8.4 Maps Activity

Αρχικά ο χρήστης κατά την είσοδο του στο Maps Activity εφόσον έχει δώσει την συγκατάθεση του στην χρήση της τοποθεσίας της συσκευής από την εφαρμογή εμφανίζεται η τρέχουσα τοποθεσία του. Η υλοποίηση του κώδικα της τρέχουσας τοποθεσίας φαίνεται στην Εικόνα 67.

```
363     private void getCurrentLocation() {
364         if (ActivityCompat.checkSelfPermission( context: this, Manifest.permission.ACCESS_FINE_LOCATION) != PackageManager.PERMISSION_GRANTED
365                 && ActivityCompat.checkSelfPermission( context: this, Manifest.permission.ACCESS_COARSE_LOCATION) != PackageManager.PERMISSION_GRANTED) {
366             ActivityCompat.requestPermissions( activity: this, new String[]{Manifest.permission.ACCESS_FINE_LOCATION}, Request_code);
367             return;
368         }
369         LocationRequest locationRequest = LocationRequest.create();
370         locationRequest.setInterval(60000);
371         locationRequest.setPriority(LocationRequest.PRIORITY_HIGH_ACCURACY);
372         locationRequest.setFastestInterval(5000);
373         LocationCallback locationCallback = onLocationResult(locationResult) → {
376
377             if (locationResult == null) {
378                 Toast.makeText(getApplicationContext(), text: "Current location is null ", Toast.LENGTH_LONG).show();
379                 return;
380             }
381             for (Location location : locationResult.getLocations()) {
382                 if (location != null) {
383                     //Toast.makeText(getApplicationContext(), "Current location is " + location.getLongitude(), Toast.LENGTH_LONG).show();
384                 }
385             }
386         };
388         fusedLocationProviderClient.requestLocationUpdates(locationRequest, locationCallback, looper: null);
389         Task<Location> task = fusedLocationProviderClient.getLastLocation();
390         task.addOnSuccessListener(new OnSuccessListener<Location>() {
391             @Override
392             public void onSuccess(@NonNull Location location) {
393                 if (location != null) {
394                     lat = location.getLatitude();
395                     lng = location.getLongitude();
396                     LatLng latLng = new LatLng(lat, lng);
397                     mMap.addMarker(new MarkerOptions().position(latLng).icon(BitmapDescriptorFactory.defaultMarker(BitmapDescriptorFactory.HUE_BLUE)).title("Τρέχου
398                     mMap.moveCamera(CameraUpdateFactory.newLatLng(latLng));
399                     mMap.moveCamera(CameraUpdateFactory.newLatLngZoom(latLng, zoom: 15));
```

Εικόνα 67: Κώδικας της τρέχουσας τοποθεσίας

Αρχικά γίνεται έλεγχος εάν ο χρήστης έχει δώσει άδεια χρήσης της τοποθεσίας από την εφαρμογή. Έπειτα δημιουργείται ένα αντικείμενο της αίτησης της τοποθεσίας και ελέγχεται εάν το αποτέλεσμα της αίτησης, δηλαδή η τοποθεσία υπάρχει ή όχι. Στη συνέχεια παίρνουμε την τελευταία τοποθεσία του χρήστη και δημιουργούμε έναν marker στον χάρτη. Στον marker αυτό δίνουμε ως παράμετρο το γεωγραφικό μήκος και πλάτος της τοποθεσίας, το χρώμα που θα έχει μπλε στην συγκεκριμένη περίπτωση και το κείμενο που εμφανίζεται εάν πατηθεί.

Επίσης για το bottom navigation που χρησιμοποιείται στο Maps Activity, στο Favorites Activity και στο Profile Activity φαίνεται στην Εικόνα 68.

```
225         BottomNavigationView bottomNavigationView = findViewById(R.id.bottom_navigation);
226         bottomNavigationView.setSelectedItemId(R.id.home); //Na einai kentriki othoni to Home
227
228         bottomNavigationView.setOnNavigationItemSelectedListener(new BottomNavigationView.OnNavigationItemSelectedListener() {
229             @Override
230             public boolean onNavigationItemSelected(@NonNull MenuItem item) {
231                 switch (item.getItemId()) {
232                     case R.id.home:
233                         return true;
234                     case R.id.favorites:
235                         startActivity(new Intent(getApplicationContext(), FavoritesActivity.class));
236                         overridePendingTransition( enterAnim: 0,  exitAnim: 0);
237                         return true;
238                     case R.id.profile:
239                         startActivity(new Intent(getApplicationContext(), ProfileActivity.class));
240                         overridePendingTransition( enterAnim: 0,  exitAnim: 0);
241                         return true;
242                 }
243                 return false;
244             }
245         });
```

Εικόνα 68: Κώδικας Bottom Navigation

Αρχικά όπως φαίνεται στην γραμμή 225 γίνεται αρχικοποίηση του. Στη συνέχεια δηλώνεται το Home ως κεντρική οθόνη, το favorites και το profile στα Favorites Activity και Profile Activity αντίστοιχα.

Έπειτα με την switch case μπορούμε να μεταφερθούμε στα αντίστοιχα Activities που θα επιλέξουμε από το bottom navigation.

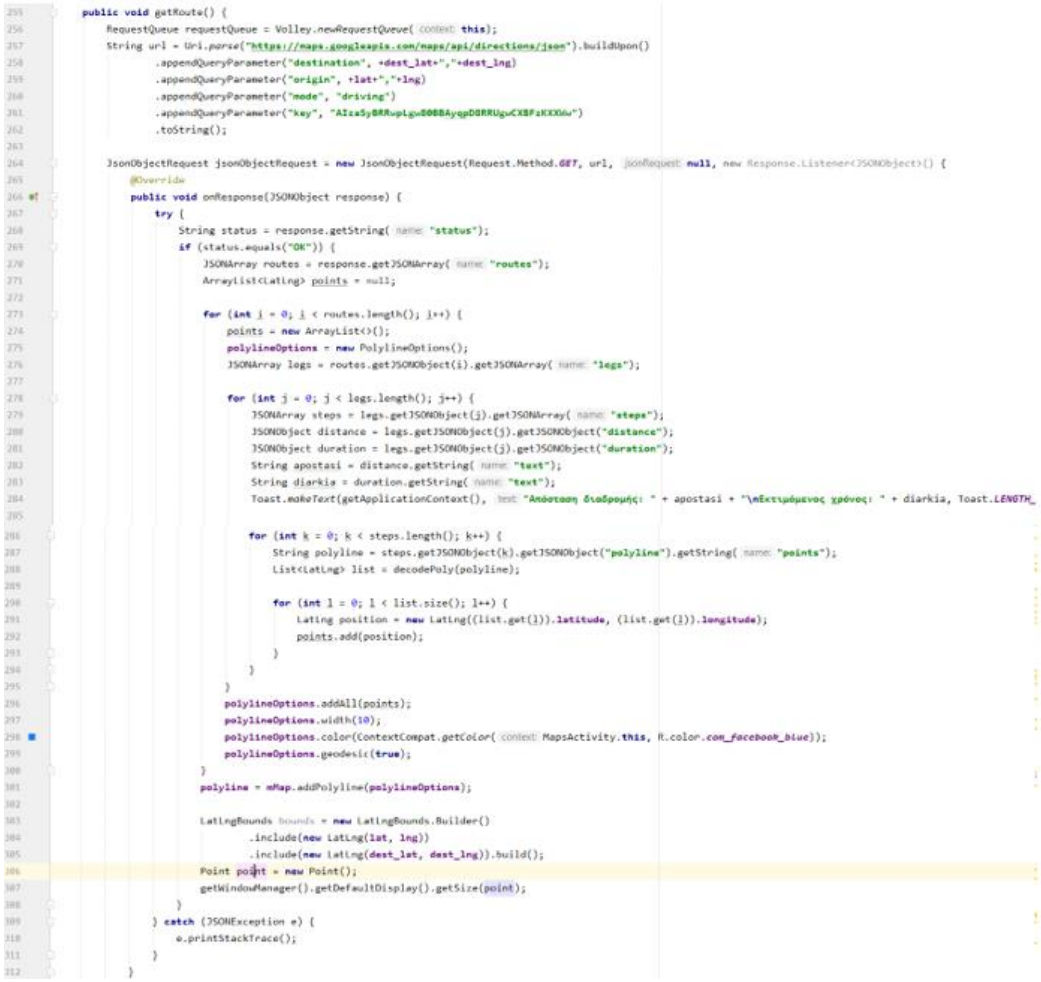
Για την εμφάνιση των σταθμών στάθμευσης στον χάρτη χρησιμοποιείται η μέθοδος Parking(). Ο κώδικας της φαίνεται στην Εικόνα 69.

```
449     public void Parking() {
450         databaseReference = FirebaseDatabase.getInstance().getReference( path: "privateParkings");
451         databaseReference.addChildEventListener(new ChildEventListener() {
452             @Override
453             public void onChildAdded(@NonNull DataSnapshot snapshot, @Nullable String previousChildName) {
454
455                 Parkings parkings = snapshot.getValue(Parkings.class);
456                 String name = parkings.getName();
457                 String adress = parkings.getAdress();
458                 String tel = parkings.getTel();
459                 int diatheseis = parkings.getDiatheseis();
460                 double price = parkings.getPrice();
461                 double lati = parkings.getLat();
462                 double lngi = parkings.getLng();
463                 LatLng location = new LatLng(lati,lngi);
464                 mMap.addMarker(new MarkerOptions().position(location).title(name).snippet("Διεύθυνση: " +adress + "\nΤηλέφωνο: " + tel +
465                         "\nΔιαθέσιμες θέσεις: " + diatheseis + "\nΤιμή: " + price + " € ανά ώρα"));
466             }
```

Εικόνα 69: Μέθοδος Parking()

Στην μέθοδο αυτή μέσα από την RealTime Database στην συλλογή privateParking δημιουργούμε ένα αντικείμενο τύπου Parkings για να ανακτήσουμε τις πληροφορίες κάθε σταθμού. Δημιουργούμε μεταβλητές για κάθε πεδίο της εγγραφής που θα χρησιμοποιηθεί για τον marker και στην συνέχεια προσθέτουμε στον χάρτη τον marker με παραμέτρους το γεωγραφικό πλάτος και μήκος, το πεδίο του ονόματος του ως τίτλο και σαν snippet, δηλαδή πληροφορίες του marker την διεύθυνση, το τηλέφωνο επικοινωνίας, τις διαθέσιμες θέσεις και την τιμή του ανά ώρα.

Για την αναζήτηση διαδρομή από την τρέχουσα τοποθεσία του προς τον σταθμό της επιλογής του χρησιμοποιείται η μέθοδος getRoute(). Ο κώδικας της μεθόδου getRoute() φαίνεται στην Εικόνα 70.



```
public void getRoute() {
    RequestQueue requestQueue = Volley.newRequestQueue( context: this);
    String url = Uri.parse("https://maps.googleapis.com/maps/api/directions/json").buildUpon()
            .appendQueryParameter("destination", +dest_lat+","+dest_lng)
            .appendQueryParameter("origin", +lat+","+lng)
            .appendQueryParameter("mode", "driving")
            .appendQueryParameter("key", "AIzaSyBRRwpLgwB0BBAyqpDORRUgwCXBFzKXXOw")
            .toString();

    JsonObjectRequest jsonObjectRequest = new JsonObjectRequest(Request.Method.GET, url,  jsonRequest null, new Response.Listener<JSONObject>() {
        @Override
        public void onResponse(JSONObject response) {
            try {
                String status = response.getString( name: "status");
                if (status.equals("OK")) {
                    JSONArray routes = response.getJSONArray( name: "routes");
                    ArrayList<LatLng> points = null;

                    for (int i = 0; i < routes.length(); i++) {
                        points = new ArrayList<>();
                        polylineOptions = new PolylineOptions();
                        JSONArray legs = routes.getJSONObject(i).getJSONArray( name: "legs");

                        for (int j = 0; j < legs.length(); j++) {
                            JSONArray steps = legs.getJSONObject(j).getJSONArray( name: "steps");
                            JSONObject distance = legs.getJSONObject(j).getJSONObject("distance");
                            JSONObject duration = legs.getJSONObject(j).getJSONObject("duration");
                            String apostasi = distance.getString( name: "text");
                            String diarkia = duration.getString( name: "text");
                            Toast.makeText(getApplicationContext(),  text: "Απόσταση διαδρομής: " + apostasi + "\nΕκτιμώμενος χρόνος: " + diarkia, Toast.LENGTH_

                            for (int k = 0; k < steps.length(); k++) {
                                String polyline = steps.getJSONObject(k).getJSONObject("polyline").getString( name: "points");
                                List<LatLng> list = decodePoly(polyline);

                                for (int l = 0; l < list.size(); l++) {
                                    LatLng position = new LatLng((list.get(l)).latitude, (list.get(l)).longitude);
                                    points.add(position);
                                }
                            }
                        }
                        polylineOptions.addAll(points);
                        polylineOptions.width(10);
                        polylineOptions.color(ContextCompat.getColor( context: MapsActivity.this, R.color.com_facebook_blue));
                        polylineOptions.geodesic(true);
                    }
                    polyline = mMap.addPolyline(polylineOptions);

                    LatLngBounds bounds = new LatLngBounds.Builder()
                            .include(new LatLng(lat, lng))
                            .include(new LatLng(dest_lat, dest_lng)).build();
                    Point point = new Point();
                    getWindowManager().getDefaultDisplay().getSize(point);
                }
            } catch (JSONException e) {
                e.printStackTrace();
            }
        }
```

Εικόνα 70: Μέθοδος getRoute()

Αρχικά δημιουργείται ένα url όπου μέσα σε αυτό περνάμε ως παραμέτρους το γεωγραφικό πλάτος και μήκος του του προορισμού και της προέλευσης, το mode, στην περίπτωση μας driving και το API key από την κονσόλα του Google Maps. Στη συνέχεια δημιουργείται ένα jsonObjectRequest όπου μέσα σε αυτό περνάμε και το url ως παράμετρο. Στο jsonObjectRequest αυτό αρχικά δημιουργείται ένας json πίνακας route και ένα ArrayList το οποίο παίρνει τιμή null αρχικά και θα περιέχει όλα τα σημεία του polyline. Στη συνέχεια δημιουργούμε τέσσερις for, στην πρώτη for δημιουργούμε το ArrayList και το polylineOptions, καθώς και ένα JSONArray που περιέχει τα legs του polyline. Στην δεύτερη for ένα JSONArray που θα περιέχει τα steps του polyline και μέσω των steps υπολογίζεται η απόσταση και η διάρκεια του polyline και εμφανίζεται με toast στον χρήστη. Στην Τρίτη for περνάμε σε ένα string όλα τα points του polyline, δηλαδή όλα τα σημεία της διαδρομής και με την χρήση της μεθόδου decodePolyline τα περνάμε σε μια λίστα τύπου LatLng. Στην τελευταία for παίρνουμε το γεωγραφικό πλάτος και το γεωγραφικό μήκος από όλα τα σημεία της διαδρομής και τα προσθέτουμε στο ArrayList points. Στη συνέχεια στα polylineOptions προσθέτουμε τα σημεία, ορίζουμε το πάχος του polyline, το χρώμα του, μπλε στην περίπτωση μας και το προσθέτουμε στον χάρτη.

Για την πλοήγηση του χρήστη στο Google Maps χρησιμοποιείται η μέθοδος navigation η οποία φαίνεται στην Εικόνα 71.

```
402     public void navigation() {
403         Uri gmmIntentUri = Uri.parse("google.navigation:q="+dest_lat+","+dest_lng);
404         Intent mapIntent = new Intent(Intent.ACTION_VIEW, gmmIntentUri);
405         mapIntent.setPackage("com.google.android.apps.maps");
406         startActivity(mapIntent);
407     }
```

Εικόνα 71: Μέθοδος navigation()

Στη μέθοδο navigation() δημιουργούμε ένα Uri όπου περνάμε το γεωγραφικό πλάτος και γεωγραφικό μήκος του προορισμού μας και μέσα ενός intent κάνουμε εκκίνηση του google maps στο τηλέφωνο μας εμφανίζοντας έτσι την διαδρομή από την τρέχουσα τοποθεσία της συσκευής στον προορισμό που δώσαμε σε μορφή GPS.

Όταν ο χρήστης πατήσει την αναζήτηση Parking το Button της πλοήγησης γίνεται αόρατο εφόσον ήταν ορατό, γίνεται καθαρισμός του χάρτη από markers και routes, καλείται η μέθοδος getCurrentLocation() για την εμφάνιση της τοποθεσίας του χρήστη και καλείται η μέθοδος Parking() για την εμφάνιση των σταθμών στάθμευσης στον χάρτη. Όταν ο χρήστης πατήσει σε κάποιον marker όπως φαίνεται και στην Εικόνα 72 το button για την πλοήγηση γίνεται αόρατο, αποθηκεύονται σε δύο μεταβλητές το γεωγραφικό πλάτος και γεωγραφικό μήκος του marker, καθαρίζεται ο χάρτης και ξανά καλούνται οι μέθοδοι getCurrentLocation() και Parking(). Στη συνέχεια υπολογίζεται η απόσταση σε ακτίνα από την τοποθεσία του χρήστη στον σταθμό που επέλεξε χωρίς δεκαδικά όπως φαίνεται 139, 140 και 142,143. Έπειτα περνάμε σε δύο μεταβλητές όπως φαίνεται στις γραμμές 145 και 146 το όνομα του σταθμού στάθμευσης και τις πληροφορίες του. Στη συνέχεια παίρνουμε δημιουργούμε τέσσερις μεταβλητές και ένα αντικείμενο τύπου UserFav το οποίο θα χρησιμοποιηθεί εάν ο χρήστης επιλέξει την αποθήκευση του σταθμού στα αγαπημένα. Τέλος δημιουργείται το Alert Dialog με τις τρεις επιλογές και σε αυτό εμφανίζεται ώς τιτλος το όνομα του σταθμού και εμφανίζονται ως πληροφορίες για τον σταθμό η διεύθυνση του, το τηλέφωνο του, οι διαθέσιμες θέσεις, η τιμή χρέωσης ανά ώρα και η απόσταση σε ακτίνα.

```
127         mMap.setOnMarkerClickListener(new GoogleMap.OnMarkerClickListener() {
128             @Override
129             public boolean onMarkerClick(@NonNull Marker marker) {
130                 navigationbtn.setVisibility(View.INVISIBLE);//otan ksanapatame allon marker na eksafanizetai
131                 dest_lat = marker.getPosition().latitude;
132                 dest_lng = marker.getPosition().longitude;
133
134                 mMap.clear();//katharismos tou xarti
135                 getCurrentLocation();//na pairnoume tin topothesia mas kai na emfanizei ton marker pou eimaste
136                 Parking(); // ksana kleisi gia na emfanizei ola ta parking
137
138                 //ypologismos tis apostasis
139                 float res[] = new float[10];
140                 Location.distanceBetween(lat, lng, dest_lat, dest_lng, res);
141                 //gia na min emfanizei dekadika
142                 DecimalFormat df = new DecimalFormat();
143                 df.setMaximumFractionDigits(0);
144
145                 String parkName = marker.getTitle().toString();
146                 String parkAdress = marker.getSnippet().toString();
147                 String parkTel = "";
148
149                 double parkLat = dest_lat;
150                 double parkLng = dest_lng;
151                 String userId = FirebaseAuth.getInstance().getCurrentUser().getUid();
152                 String id = marker.getTitle()+""+ FirebaseAuth.getInstance().getCurrentUser().getUid();
153                 UserFav user = new UserFav(parkName, parkAdress, parkTel, parkLat, parkLng, userId);
154                 //-------------------------------------------------------------------------
155
156                 AlertDialog.Builder infoParkingDialog = new AlertDialog.Builder( context: MapsActivity.this);
157                 infoParkingDialog.setTitle("" + marker.getTitle());
158                 infoParkingDialog.setMessage("" + marker.getSnippet() + "\nΑπόσταση σε ακτίνα: " + df.format(res[0]) + "μέτρα");
```

Εικόνα 72: Κώδικας εμφάνισης πληροφορίες σταθμού

Ο κώδικας των τριών επιλογών του Alert Dialog φαίνεται στην Εικόνα 73.

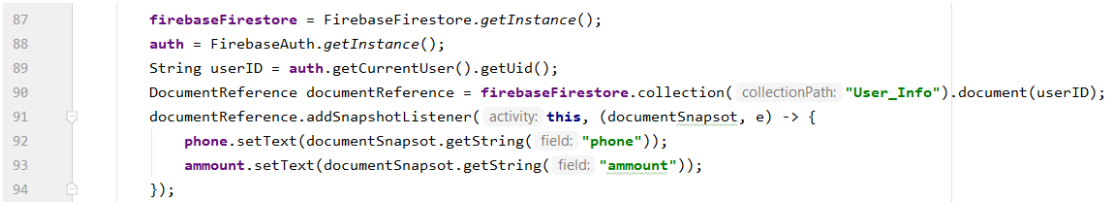
```
infoParkingDialog.setPositiveButton( text: "Αναζήτηση Διαδρομής", new DialogInterface.OnClickListener() {
    @Override
    public void onClick(DialogInterface dialog, int which) {
        getRoute(); //Klisi tis methodou gia euresi tis syntomoteris diadromis
        navigationbtn.setVisibility(View.VISIBLE); //emfanisi tou buttom gia ploigisi
        navigationbtn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) { navigation(); }
        });
    }
});
infoParkingDialog.setNeutralButton( text: "Κάντε κράτηση", new DialogInterface.OnClickListener() {
    @Override
    public void onClick(DialogInterface dialog, int which) {
        Intent intentBook = new Intent(getApplicationContext(), BookingActivity.class);
        intentBook.putExtra( name: "parking_name", marker.getTitle().toString());
        startActivity(intentBook);
    }
});
infoParkingDialog.setNegativeButton( text: "Προσθήκη στα αγαπημένα", new DialogInterface.OnClickListener() {
    @Override
    public void onClick(DialogInterface dialog, int which) {
        databaseReference = FirebaseDatabase.getInstance().getReference( path: "parkings");
        databaseReference./*child("parkings").*/child(id).setValue(user).addOnCompleteListener(new OnCompleteListener<Void>() {
            @Override
            public void onComplete(@NonNull Task<Void> task) {
                if (task.isSuccessful()) {
                    Toast.makeText(getApplicationContext(), text: "Αποθήκευση στα αγαπημένα...", Toast.LENGTH_SHORT).show();
                } else {
                    Toast.makeText(getApplicationContext(), text: "Κατι πηγε στραβα", Toast.LENGTH_SHORT).show();
                }
            }
        });
    }
});
```

Εικόνα 73: Κώδικας των επιλεγών για τον σταθμό

Για την επιλογή της αναζήτησης διαδρομής καλείται η μέθοδος getRoute() για την εμφάνιση του polyline της διαδρομής στον χάρτη και το Button για την πλοήγηση γίνεται ορατό. Εφόσον ο χρήστης επιλέξει την πλοήγηση καλείται η μέθοδος navigation(). Για την επιλογή της κράτησης γίνεται εκκίνηση του Booking Activity και επίσης περνάμε σαν πληροφορία σε αυτό το όνομα του σταθμού. Για την επιλογή της προσθήκης στα αγαπημένα αποθηκεύεται στην συλλογή parkings μια νέα εγγραφή με τίτλο το όνομα του σταθμού και το userId του χρήστη και στα πεδία του αποθηκεύονται οι τιμές που περάσαμε στο αντικείμενο UserFav.

3.8.5 Booking Activity

Το Booking Activity χρησιμοποιείται για την υλοποίηση κράτησης θέσης στάθμευσης από τον χρήστη. Αρχικά θέτουμε στο text view στην κορυφή της εφαρμογής το οποίο έχει περαστεί μέσω του intent από το Alert Dialog του Maps Activity. Τα δύο πρώτα text views εμφανίζουν το τηλέφωνο και το διαθέσιμο υπόλοιπο του χρήστη τα οποία τα παίρνουμε από την βάση δεδομένων όπως φαίνεται στην Εικόνα 74.



```
firebaseFirestore = FirebaseFirestore.getInstance();
auth = FirebaseAuth.getInstance();
String userID = auth.getCurrentUser().getUid();
DocumentReference documentReference = firebaseFirestore.collection( collectionPath: "User_Info").document(userID);
documentReference.addSnapshotListener( activity: this, (documentSnapsot, e) -> {
    phone.setText(documentSnapsot.getString( field: "phone"));
    ammount.setText(documentSnapsot.getString( field: "ammount"));
});
```

Εικόνα 74: Διαθέσιμο ποσό και τηλέφωνο του χρήστη.

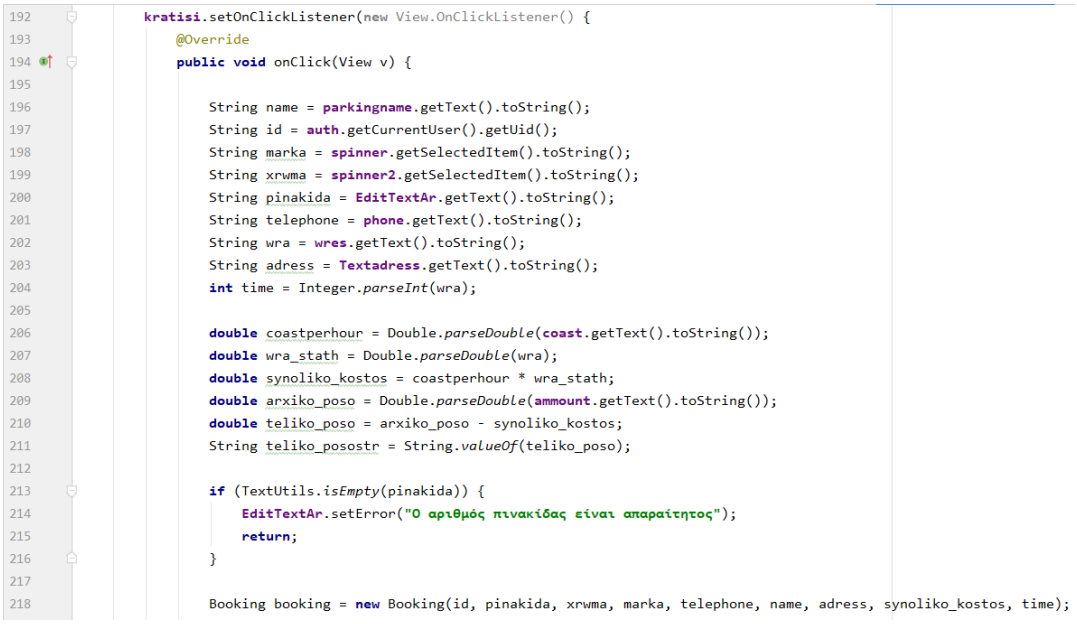
Στα επόμενα τέσσερα text view εμφανίζονται κάποια στοιχεία του σταθμού στάθμευσης όπως η διεύθυνση, οι διαθέσιμες θέσεις, το τηλέφωνο του σταθμού και το κόστος στάθμευσης ανά ώρα. Για να εμφανίσουμε αυτές τις πληροφορίες τις ανακτούμε από την βάση δεδομένων στην συλλογή των private parkings όπως φαίνεται στην Εικόνα 75.

```
128         databaseReference = FirebaseDatabase.getInstance().getReference( path: "privateParkings");
129         databaseReference.addChildEventListener(new ChildEventListener() {
130             @Override
131             public void onChildAdded(@NonNull DataSnapshot snapshot, @Nullable String previousChildName) {
132                 Parkings parkings = snapshot.getValue(Parkings.class);
133                 String name = parkings.getName();
134                 String adress = parkings.getAdress();
135                 String tel = parkings.getTel();
136                 int thesis = parkings.getTheseis();
137                 int diathesimes = parkings.getDiatheseis();
138                 double price = parkings.getPrice();
139                 double lati = parkings.getLat();
140                 double lngi = parkings.getLng();
141                 String id = parkings.getOwnerId();
142
143                 try {
144                     if (name.equals(parkingname.getText().toString())) {
145                         Textadress.setText(adress);
146                         available.setText("" + diathesimes);
147                         avalaibleseats.setText("" + diathesimes + " διαθέσιμες θέσεις");
148                         parkingphone.setText(tel);
149                         parkincoast.setText("" + price + " ανά ώρα");
150                         coast.setText("" + price);
151                         destlat.setText("" + lati);
152                         destlng.setText("" + lngi);
153                     }
154                 } catch (Exception e) {
155                     e.printStackTrace();
156                 }
157             }
```

Εικόνα 75: Εμφάνιση πληροφοριών του σταθμού.

Επίσης πρέπει να γίνει έλεγχος όπως φαίνεται στην γραμμή 144 της παραπάνω Εικόνας. Ουσιαστικά ελέγχεται εάν το όνομα του σταθμού στάθμευσης από την βάση δεδομένων είναι ίδιο με το text view στην κορυφή της διεπαφής. Αν είναι ίδιο τότε εμφανίζονται τα στοιχεία του συγκεκριμένου σταθμού. Ο έλεγχος αυτός είναι απαραίτητος καθώς εάν δεν γίνει εμφανίζονται τα στοιχεία του σταθμού της πρώτης εγγραφής από την συλλογή private parkings.

Στη συνέχεια ο χρήστης πρέπει να συμπληρώσει τα στοιχεία του και εφόσον τα συμπληρώσει μπορεί να κάνει κράτηση για θέση στάθμευσης. Ο τρόπος υλοποίησης της διαδικασίας κράτησης φαίνεται από τις παρακάτω Εικόνες.

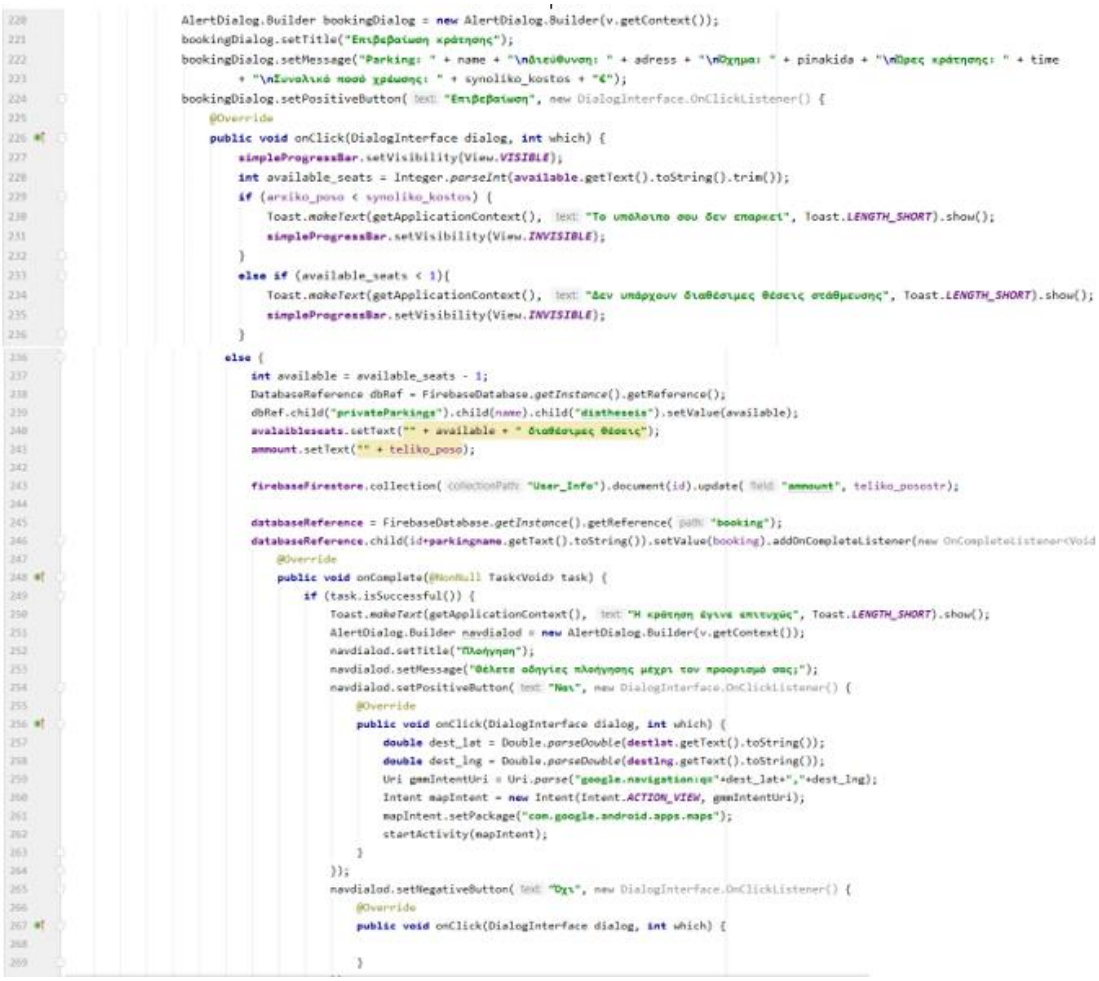


```
192        kratisi.setOnClickListener(new View.OnClickListener() {
193            @Override
194            public void onClick(View v) {
195
196                String name = parkingname.getText().toString();
197                String id = auth.getCurrentUser().getUid();
198                String marka = spinner.getSelectedItem().toString();
199                String xrwma = spinner2.getSelectedItem().toString();
200                String pinakida = EditTextAr.getText().toString();
201                String telephone = phone.getText().toString();
202                String wra = wres.getText().toString();
203                String adress = Textadress.getText().toString();
204                int time = Integer.parseInt(wra);
205
206                double coastperhour = Double.parseDouble(coast.getText().toString());
207                double wra_stath = Double.parseDouble(wra);
208                double synoliko_kostos = coastperhour * wra_stath;
209                double arxiko_poso = Double.parseDouble(ammount.getText().toString());
210                double teliko_poso = arxiko_poso - synoliko_kostos;
211                String teliko_pososTr = String.valueOf(teliko_poso);
212
213                if (TextUtils.isEmpty(pinakida)) {
214                    EditTextAr.setError("Ο αριθμός πινακίδας είναι απαραίτητος");
215                    return;
216                }
217
218                Booking booking = new Booking(id, pinakida, xrwma, marka, telephone, name, adress, synoliko_kostos, time);
```

Εικόνα 76: Υλοποίηση κράτησης (α)

Όπως φαίνεται στην παραπάνω Εικόνα αρχικά δημιουργούνται μεταβλητές για την αποθήκευση των πληροφοριών στου σταθμού στάθμευσης, του χρήστη και του οχήματος του χρήστη. Στη συνέχεια όπως φαίνεται στη γραμμή 208 υπολογίζεται το συνολικό ποσό της συναλλαγής πολλαπλασιάζοντας το κόστος ανά ώρα με τις ώρες που έχει επιλέξει ο χρήστης να είναι η κράτηση του, καθώς και στην γραμμή 210 γίνεται ο υπολογισμός του ποσού που απομένει στον χρήστη αφαιρώντας από το ποσό του το συνολικό κόστος της κράτησης. Γίνεται έλεγχος εάν το edit text του αριθμού κυκλοφορίας είναι συμπληρωμένο και στην συνέχεια δημιουργείται ένα αντικείμενο τύπου Booking.

Στη συνέχεια όπως φαίνεται και στην Εικόνα δημιουργείται το Alert Dialog που προειδοποιεί τον χρήστη εάν επιθυμεί να κάνει την κράτηση. Στο Alert Dialog εμφανίζονται πληροφορίες για την κράτηση όπως το όνομα του σταθμού, η διεύθυνση του, η πινακίδα του οχήματος του χρήστη, οι ώρες κρατήσεις που έχει επιλέξει ο χρήστης και το συνολικό κόστος της κράτησης.



```
220             AlertDialog.Builder bookingDialog = new AlertDialog.Builder(v.getContext());
221             bookingDialog.setTitle("Επιβεβαίωση κράτησης");
222             bookingDialog.setMessage("Parking: " + name + "\nδιεύθυνση: " + adress + "\nΟχημα: " + pinakida + "\nΩρες κράτησης: " + time
223                     + "\nΣυνολικό ποσό χρέωσης: " + synoliko_kostos + "€");
224             bookingDialog.setPositiveButton( text: "Επιβεβαίωση", new DialogInterface.OnClickListener() {
225                 @Override
226                 public void onClick(DialogInterface dialog, int which) {
227                     simpleProgressBar.setVisibility(View.VISIBLE);
228                     int available_seats = Integer.parseInt(available.getText().toString().trim());
229                     if (arxiko_poso < synoliko_kostos) {
230                         Toast.makeText(getApplicationContext(), text: "Το υπόλοιπο σου δεν επαρκεί", Toast.LENGTH_SHORT).show();
231                         simpleProgressBar.setVisibility(View.INVISIBLE);
232                     }
233                     else if (available_seats < 1){
234                         Toast.makeText(getApplicationContext(), text: "Δεν υπάρχουν διαθέσιμες θέσεις στάθμευσης", Toast.LENGTH_SHORT).show();
235                         simpleProgressBar.setVisibility(View.INVISIBLE);
236                     }
237                     else {
238                         int available = available_seats - 1;
239                         DatabaseReference dbRef = FirebaseDatabase.getInstance().getReference();
240                         dbRef.child("privateParkings").child(name).child("diatheseis").setValue(available);
241                         avalaibleseats.setText("" + available + " διαθέσιμες θέσεις");
242                         amnount.setText("" + teliko_poso);
243
244                         firebaseFirestore.collection( collectionPath: "User_Info").document(id).update( field: "amnount", teliko_posostr);
245
246                         databaseReference = FirebaseDatabase.getInstance().getReference( path: "booking");
247                         databaseReference.child(id+parkingname.getText().toString()).setValue(booking).addOnCompleteListener(new OnCompleteListener<Void
248                             @Override
249                             public void onComplete(@NonNull Task<Void> task) {
250                                 if (task.isSuccessful()) {
251                                     Toast.makeText(getApplicationContext(), text: "Η κράτηση έγινε επιτυχώς", Toast.LENGTH_SHORT).show();
252                                     AlertDialog.Builder navdialod = new AlertDialog.Builder(v.getContext());
253                                     navdialod.setTitle("Πλοήγηση");
254                                     navdialod.setMessage("Θέλετε οδηγίες πλοήγησης μέχρι τον προορισμό σας;");
255                                     navdialod.setPositiveButton( text: "Ναι", new DialogInterface.OnClickListener() {
256                                         @Override
257                                         public void onClick(DialogInterface dialog, int which) {
258                                             double dest_lat = Double.parseDouble(destlat.getText().toString());
259                                             double dest_lng = Double.parseDouble(destlng.getText().toString());
260                                             Uri gmmIntentUri = Uri.parse("google.navigation:q="+dest_lat+","+dest_lng);
261                                             Intent mapIntent = new Intent(Intent.ACTION_VIEW, gmmIntentUri);
262                                             mapIntent.setPackage("com.google.android.apps.maps");
263                                             startActivity(mapIntent);
264                                         }
265                                     });
266                                     navdialod.setNegativeButton( text: "Οχι", new DialogInterface.OnClickListener() {
267                                         @Override
268                                         public void onClick(DialogInterface dialog, int which) {
269
```

Εικόνα 77: Υλοποίηση κράτησης (β)

Πατώντας την επιλογή της επιβεβαίωσης της κράτησης όπως φαίνεται στην γραμμή 228 γίνεται έλεγχος εάν ο χρήστης έχει το απαραίτητο διαθέσιμο ποσό και στην γραμμή 233 εάν ο σταθμός στάθμευσης έχει τουλάχιστον μία διαθέσιμη θέση στάθμευσης.

Εάν ο χρήστης διαθέτει το ποσό και ο σταθμός στάθμευσης έχει τουλάχιστον μία διαθέσιμη θέση όπως φαίνεται στην παρακάτω Εικόνα στην γραμμή 237 οι διαθέσιμες θέσεις μειώνονται κατά ένα, στην βάση δεδομένων στην εγγραφή του συγκεκριμένου σταθμού το πεδίο των διαθέσιμων θέσεων παίρνει την νέα τιμή, καθώς το νέο διαθέσιμο ποσό αντικαθιστά το παλιό στο αντίστοιχο text view και γίνεται update στο πεδίο του διαθέσιμου ποσού στην εγγραφή του χρήστη.

Στην συνέχεια όπως φαίνεται στις γραμμές 245 και 256 δημιουργείται μία νέα εγγραφή στη συλλογή bookings που περιέχει τα στοιχεία της κράτησης. Εφόσον η νέα εγγραφή δημιουργηθεί εμφανίζεται μήνυμα στον χρήστη και δημιουργείται ένα Alert Dialog που ρωτάει τον χρήστη εάν θέλει πλοήγηση έως τον σταθμό.

3.8.6 Favorites Activity & Kratiseis Activity

Το Favorites Activity και το Kratiseis Activity υλοποιήθηκαν με τον ίδιο τρόπο. Στην Εικόνα 78 φαίνεται ο κώδικας της μεθόδου onCreate που είναι ίδια και στα δύο Activity.

```
43          recyclerView = findViewById(R.id.recycleview);
44          databaseReference = FirebaseDatabase.getInstance().getReference( path: "parkings");
45          list = new ArrayList<>();
46          recyclerView.setLayoutManager(new LinearLayoutManager( context: this));
47          adapter = new MyAdapter( context: this, list);
48          recyclerView.setAdapter(adapter);
49
50          databaseReference.addValueEventListener(new ValueEventListener() {
51              @Override
52              public void onDataChange(@NonNull DataSnapshot snapshot) {
53                  for (DataSnapshot dataSnapshot : snapshot.getChildren()) {
54                      UserFav user = dataSnapshot.getValue(UserFav.class);
55                      list.add(user);
56                  }
57                  adapter.notifyDataSetChanged();
58              }
59              @Override
60              public void onCancelled(@NonNull DatabaseError error) {
61
62              }
63          });
```

Εικόνα 78: onCreate Favorites και Kratiseis Activity

Όπως φαίνεται στην γραμμή 43 αρχικά γίνεται αρχικοποίηση του recyclerView. Στην γραμμή 48 θέτουμε adapter στο recyclerView το οποίο είναι τύπου MyAdapter για το favorites Activity και MyAdapterBooking για το Kratiseis Activity. Έπειτα μέσα από τη βάση δεδομένων με την βοήθεια ενός αντικειμένου τύπου UseFav παίρνουμε τις πληροφορίες του αγαπημένου σταθμού και τις προσθέτουμε στο ArrayList που έχουμε δημιουργήσει. Αντίστοιχα για το Kratiseis Activity χρησιμοποιούμε ένα αντικείμενο τύπου Booking για να πάρουμε τις πληροφορίες της αντίστοιχης κράτησης από τη βάση δεδομένων.

3.8.7 Profile Activity

Όπως είδαμε και στο προηγούμενο κεφάλαιο το Profile Activity αποτελείται από ένα Image View που περιέχει την εικόνα προφίλ του χρήστη, δύο text view με τις πληροφορίες του και έξι material button. Τα δύο τελευταία material button για την αναφορά προβλήματος και τις πληροφορίες εφαρμογής περιέχουν από ένα Alert Dialog. Αρχικά στην μέθοδο onCreate του Activity καλείται η μέθοδος checkUser() στην γραμμή 59 της Εικόνας 79. Στην συνέχεια μέσα σε μία try catch με την χρήση μιας for ελέγχουμε με τι τρόπο είναι συνδεδεμένος ο χρήστης στην εφαρμογή. Στη συνέχεια γίνεται έλεγχος αν ο χρήστης είναι συνδεδεμένος με google ή github. Εάν ναι τότε δεν υπάρχει δυνατότητα για την αλλαγή κωδικού πρόσβασης και email του χρήστη. Αν ο χρήστης είναι συνδεδεμένος με email και password τότε πατώντας το αντίστοιχο material button γίνεται εκκίνηση του ChangeEmail Activity ή του ChangePassword Activity ανάλογα με το ποιο θα επιλέξει ο χρήστης.

```
59          checkUser();
60
61          //gia tin allagi email kai password (kanw prwta elegxo an einai syndedemenos me google i github)
62          try {
63              for (UserInfo user : FirebaseAuth.getInstance().getCurrentUser().getProviderData()) {
64                  binding.changeEmail.setOnClickListener(new View.OnClickListener() {
65                      @Override
66                      public void onClick(View v) {
67                          if (user.getProviderId().equals("github.com") || user.getProviderId().equals("google.com"))  {
68                              Toast.makeText(getApplicationContext(),  text: "Δεν υποστηρίζεται αυτή η λειτουργία", Toast.LENGTH_SHORT).show();
69                          } else {
70                              startActivity(new Intent(getApplicationContext(), ChangeEmailActivity.class));
71                          }
72                      }
73                  });
74
75                  binding.changePass.setOnClickListener(new View.OnClickListener() {
76                      @Override
77                      public void onClick(View v) {
78                          if (user.getProviderId().equals("github.com") || user.getProviderId().equals("google.com")) {
79                              Toast.makeText(getApplicationContext(),  text: "Δεν υποστηρίζεται αυτή η λειτουργία", Toast.LENGTH_SHORT).show();
80                          } else {
81                              startActivity(new Intent(getApplicationContext(), ChangePasswordActivity.class));
82                          }
83                      }
84                  });
85              }
86          } catch (Exception e) {
87              e.printStackTrace();
```

Εικόνα 79: onCreate() Profile Activity

Η μέθοδος checkUser() χρησιμοποιείται για να εμφανίζονται τα στοιχεία του χρήστη στο Profile Activity.

```
232     private void checkUser() {
233         FirebaseUser firebaseUser = auth.getCurrentUser();
234         firebaseFirestore = FirebaseFirestore.getInstance();
235         if (firebaseUser == null) {
236             startActivity(new Intent( packageContext: this, LoginActivity.class));
237             finish();
238         }
239         else {
240             String email = firebaseUser.getEmail();
241             String name = firebaseUser.getDisplayName();
242             String tel = firebaseUser.getPhoneNumber();
243             String userID = auth.getCurrentUser().getUid();
244             DocumentReference documentReference = firebaseFirestore.collection( collectionPath: "User_Info").document(userID);
245
246             for (UserInfo user: FirebaseAuth.getInstance().getCurrentUser().getProviderData()) {
247                 if (user.getProviderId().equals("github.com") || user.getProviderId().equals("google.com")) {
248                     binding.emailTv.setText(email);
249                     binding.nameTv.setText(name);
250                     Glide.with( activity: ProfileActivity.this).load(firebaseUser.getPhotoUrl()).override(190).circleCrop().into(userpic);
251                 }
252                 else {
253                     binding.emailTv.setText(email);
254                     documentReference.addSnapshotListener( activity: this, (documentSnapsot, e) -> {
255                         nameTV.setText(documentSnapsot.getString( field: "name"));
256                     });
257
258                 }
259             }
260         }
261     }
```

Εικόνα 80: Μέθοδος checkUser()

Αρχικά στην μέθοδο checkUser() γίνεται έλεγχος εάν ο χρήστης είναι συνδεδεμένος στην βάση δεδομένων. Αν είναι, στη συνέχεια ελέγχουμε αν ο χρήστης είναι συνδεδεμένος με email και password ή είναι συνδεδεμένος με google ή github. Εάν ο χρήστης είναι συνδεδεμένος με github ή google τότε παίρνουμε από τον χρήστη της firebase την εικόνα προφίλ, το email και το όνομα και τα τοποθετούμε στα αντίστοιχα πεδία. Αν ο χρήστης είναι συνδεδεμένος με email και password τότε παίρνουμε από το firestore τα πεδία του ονόματος και του email από το αντίστοιχο document του χρήστη που είναι συνδεδεμένος και τα τοποθετούμε στα αντίστοιχα πεδία.

Για την αποσύνδεση του χρήστη και την διαγραφεί του λογαριασμού του η υλοποίηση είναι η ίδια με την αποσύνδεση του διαχειριστή και την διαγραφή του σταθμού από την εφαρμογή στο ParkingOwner Activity και θα αναλυθεί στην υποενότητα 3.8.9.

3.8.8 ChangeEmail Activity & ChangePassword Activity

Η αλλαγή του email και του κωδικού πρόσβασης του χρήστη υλοποιήθηκε με τον ίδιο τρόπο. Για την αλλαγή του email αφού γίνει αρχικοποίηση των μεταβλητών πατώντας το Change Email γίνεται κλήση της μεθόδου changeEmail. Η μέθοδος changeEmail φαίνεται στην Εικόνα 81.

```
51    private void changeEmail() {
52        if (email.getText().toString().isEmpty() && password.getText().toString().isEmpty() && change.getText().toString().isEmpty()) {
53            Toast.makeText( context: ChangeEmailActivity.this,  text: "Συμπληρώστε όλα τα παιδία", Toast.LENGTH_LONG).show();
54        }
55        else {
56            user = auth.getCurrentUser();
57
58            if (user != null && user.getEmail() != null) {
59                AuthCredential credential = EmailAuthProvider.getCredential(user.getEmail(), password.getText().toString());
60                user.reauthenticate(credential).addOnCompleteListener(new OnCompleteListener<Void>() {
61                    @Override
62                    public void onComplete(@NonNull Task<Void> task) {
63                        if (task.isSuccessful()) {
64                            progressBar.setVisibility(View.VISIBLE);
65                            //Toast.makeText(ChangeEmailActivity.this, "Re authentication is success", Toast.LENGTH_LONG).show();
66                            user.updateEmail(change.getText().toString()).addOnCompleteListener(new OnCompleteListener<Void>() {
67                                @Override
68                                public void onComplete(@NonNull Task<Void> task) {
69                                    Toast.makeText( context: ChangeEmailActivity.this,  text: "Το email άλλαξε", Toast.LENGTH_LONG).show();
70                                    auth.signOut();
71                                    Intent i = new Intent( packageContext: ChangeEmailActivity.this, LoginActivity.class);
72                                    startActivity(i);
73                                    finish();
74                                }
75                            });
76                        }
77                        else {
78                            Toast.makeText( context: ChangeEmailActivity.this,  text: "Re authentication is failed", Toast.LENGTH_LONG).show();
79                        }
80                    }
81                });
82            }
83            else {
84                Intent i = new Intent( packageContext: ChangeEmailActivity.this, LoginActivity.class);
85                startActivity(i);
```

Εικόνα 81: Μέθεδος για την αλλαγή του email

Αρχικά γίνεται έλεγχος εάν τα πεδία του email, του κωδικού πρόσβασης και του καινούργιου email είναι συμπληρωμένα. Αν έχουν συμπληρωθεί ελέγχεται εάν υπάρχει χρήστης και αν υπάρχει το email του χρήστη γίνεται κλήση της μεθόδου reauthenticate που δέχεται ως παράμετρο το email και τον κωδικό πρόσβασης του χρήση. Αν η κλήση της μεθόδου γίνει με επιτυχία καλείται η μέθοδος updateEmail που δέχεται ως παράμετρο το νέο email που έχει δώσει ο χρήστης. Όταν ολοκληρωθεί η διαδικασία της αλλαγής του email γίνεται αποσύνδεση του χρήστη από τη βάση δεδομένων και μεταφέρεται στο Login Activity.

Για την αλλαγή του κωδικού πρόσβασης αφού γίνει αρχικοποίηση των μεταβλητών πατώντας το Change Password καλείται η μέθοδος changePassword. Η μέθοδος changePassword φαίνεται στην Εικόνα 82.

```
51  private void changePassword() {
52      if (pass.getText().toString().isEmpty() && newpass.getText().toString().isEmpty() && confirm.getText().toString().isEmpty()) {
53          Toast.makeText( context: ChangePasswordActivity.this,  text: "Συμπληρώστε όλα τα παιδία", Toast.LENGTH_LONG).show();
54      }
55      else  {
56          if (newpass.getText().toString().equals(confirm.getText().toString())) {
57              user = auth.getCurrentUser();
58
59              if (user != null && user.getEmail() != null) {
60                  AuthCredential credential = EmailAuthProvider.getCredential(user.getEmail(), pass.getText().toString());
61                  user.reauthenticate(credential).addOnCompleteListener(new OnCompleteListener<Void>() {
62                      @Override
63                      public void onComplete(@NonNull Task<Void> task) {
64                          if (task.isSuccessful()) {
65                              progressBar.setVisibility(View.VISIBLE);
66                              //Toast.makeText(ChangePasswordActivity.this, "Re authentication is success", Toast.LENGTH_LONG).show();
67                              user.updatePassword(newpass.getText().toString()).addOnCompleteListener(new OnCompleteListener<Void>() {
68                                  @Override
69                                  public void onComplete(@NonNull Task<Void> task) {
70                                      Toast.makeText( context: ChangePasswordActivity.this,  text: "Ο κωδικός άλλαξε", Toast.LENGTH_LONG).show();
71                                      auth.signOut();
72                                      Intent i = new Intent( packageContext: ChangePasswordActivity.this, LoginActivity.class);startActivity(i);
73                                      finish();
74                                  }
75                              });
76                          }
77                          else {
78                              Toast.makeText( context: ChangePasswordActivity.this,  text: "Re authentication is failed", Toast.LENGTH_LONG).show();
79                          }
80                      }
81                  });
82              }
83              else {
84                  Intent i = new Intent( packageContext: ChangePasswordActivity.this, LoginActivity.class);
85                  startActivity(i);
```

Εικόνα 82: Μέθοδος για την αλλαγή του κωδικού πρόσβασης

Όπως και στην μέθοδο changeEmail έτσι και εδώ γίνεται έλεγχος εάν τα πεδία του κωδικού πρόσβασης, του νέου κωδικού πρόσβασης και της επιβεβαίωσης του νέου κωδικού πρόσβασης είναι συμπληρωμένα. Αν έχουν συμπληρωθεί ελέγχεται αν ο νέος κωδικός είναι ίδιος με την επιβεβαίωση του νέου κωδικού είναι ίδια. Μετά γίνεται έλεγχος για το εάν υπάρχει χρήστης και αν υπάρχει το email του χρήστη και γίνεται κλήση της μεθόδου reauthenticate που δέχεται ως παράμετρο το email και τον κωδικό πρόσβασης του χρήση. Αν η κλήση της μεθόδου γίνει με επιτυχία καλείται η μέθοδος updatePassword που δέχεται ως παράμετρο τον νέο κωδικό πρόσβασης που έχει δώσει ο χρήστης. Όταν ολοκληρωθεί η διαδικασία της αλλαγής του κωδικού πρόσβασης γίνεται αποσύνδεση του χρήστη από τη βάση δεδομένων και μεταφέρεται στο Login Activity.

3.8.9 ParkingOwner Activity

Το τελευταίο Activity που θα παρουσιαστεί είναι το ParkingOwner Activity. Στο Activity αυτό αρχικά υπάρχουν δύο material button. Το πρώτο material button χρησιμοποιείται για την διαγραφή του σταθμού στάθμευσης από την εφαρμογή. Ο κώδικας υλοποίησης της εφαρμογής φαίνεται στην Εικόνα 83.

Πατώντας την διαγραφή ο διαχειριστής δημιουργείται ένα Alert Dialog που τον προειδοποιεί για την διαγραφή του σταθμού. Αφού ο διαχειριστής επιλέξει την διαγραφή όπως φαίνεται και στην γραμμή 181 γίνεται χρήση της μεθόδου removeValue για την διαγραφή της εγγραφής του συγκεκριμένου σταθμού. Εφόσον η διαγραφή είναι επιτυχής διαγράφεται και ο διαχειριστής από την εφαρμογή με την χρήση της user.delete().

```
binding.deleteParkingBtn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        AlertDialog.Builder deleteDialog = new AlertDialog.Builder(v.getContext());
        deleteDialog.setTitle("Διαγραφή Parking");
        deleteDialog.setMessage("Είστε σίγουρος/η οτι θέλετε να διαγράψετε τον λογαριασμό Parking σας;");
        deleteDialog.setPositiveButton( text: "Ναι", new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
                databaseReference = FirebaseDatabase.getInstance().getReference( path: "privateParkings");
                String parkingName = parkingname.getText().toString();
                FirebaseUser user = FirebaseAuth.getInstance().getCurrentUser();
                databaseReference.child(parkingName).removeValue().addOnCompleteListener(new OnCompleteListener<Void>() {
                    @Override
                    public void onComplete(@NonNull Task<Void> task) {
                        if (task.isSuccessful()) {
                            user.delete().addOnCompleteListener(new OnCompleteListener<Void>() {
                                @Override
                                public void onComplete(@NonNull Task<Void> task) {
                                    if (task.isSuccessful()) {
                                        Toast.makeText(getApplicationContext(), text: "Ο λογαριασμός διεφράφει!", Toast.LENGTH_SHORT).show();
                                        Intent i = new Intent( packageContext: ParkingOwnerActivity.this, ParkingOwnerSignInActivity.class);
                                        startActivity(i);
                                    } else {
                                        Toast.makeText(getApplicationContext(), text: "Account not deleted!", Toast.LENGTH_SHORT).show();
                                    }
                                }
                            });
                        } else {
                            Toast.makeText(getApplicationContext(), text: "Ανεπιτυχής διαφραφή" , Toast.LENGTH_LONG).show();
                        }
                    }
                });
            }
```

Εικόνα 83: Διαγραφή του σταθμού στάθμευσης

Το δεύτερο είναι για την αποσύνδεση του διαχειριστή από τον λογαριασμό και ο κώδικας του παρουσιάζεται στην Εικόνα 84.

```
binding.logoutParkingBtn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        AlertDialog.Builder logoutDialog = new AlertDialog.Builder(v.getContext());
        logoutDialog.setTitle("Αποσύνδεση");
        logoutDialog.setMessage("Είστε σίγουρος/η οτι θέλετε να αποσυνδεθείτε;");
        logoutDialog.setPositiveButton( text: "Ναι", new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
                FirebaseAuth.getInstance().signOut();
                startActivity(new Intent(getApplicationContext(), ParkingOwnerSignInActivity.class));
                finish();
            }
        });
        logoutDialog.setNegativeButton( text: "Όχι", new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {

            }
        });
        logoutDialog.create().show();
    }
});
```

Εικόνα 84: Αποσύνδεση του διαχειριστή

Όπως και στην διαγραφή υπάρχει ένα Alert Dialog που προειδοποιεί τον διαχειριστή για την αποσύνδεση του από την εφαρμογή. Αφού ο διαχειριστής επιλέξει να αποσυνδεθεί γίνεται αποσύνδεση από την βάση δεδομένων όπως φαίνεται στην γραμμή 223 και γίνεται εκκίνηση του ParkingOwnerSignIn Activity.

Στα πεδία εμφανίζονται όλες οι πληροφορίες του σταθμού από την βάση δεδομένων. Για την αύξηση και την μείωση των διαθέσιμων θέσεων χρησιμοποιήθηκαν δύο image buttons και ένα text view. Η υλοποίηση της συγκεκριμένης λειτουργίας φαίνεται στην παρακάτω Εικόνα 85.

```
89          plusButton.setOnClickListener(new View.OnClickListener() {
90              @Override
91              public void onClick(View v) {
92                  String theseis = diatheseis.getText().toString();
93                  int thesi = Integer.parseInt(theseis);
94                  thesi++;
95                  theseis = String.valueOf(thesi);
96                  diatheseis.setText(theseis);
97              }
98          });
99
100         minusButton.setOnClickListener(new View.OnClickListener() {
101             @Override
102             public void onClick(View v) {
103                 String theseis = diatheseis.getText().toString();
104                 int thesi = Integer.parseInt(theseis);
105                 if (thesi > 0) {
106                     thesi--;
107                     theseis = String.valueOf(thesi);
108                     diatheseis.setText(theseis);
109                 }
110             }
111         });
```

Εικόνα 85: Αυξομείωση διαθέσιμων θέσεων σταθμού.

Σε κάθε πάτημα του image button όπως φαίνεται στις γραμμές 92 και 93 καθώς και στις 103 και 104 παίρνουμε την String τιμή του text view και την μετατρέπουμε σε integer. Στη συνέχεια αυξάνεται κατά ένα ή μειώνεται κατά ένα εξαρτάται ποιο image button θα επιλέξει ο διαχειριστής και η καινούργια τιμή μετατρέπεται σε String και τοποθετείται στο text view. Στην περίπτωση της μείωσης θέσεων γίνεται έλεγχος εάν οι θέσεις είναι μεγαλύτερες από το μηδέν ώστε να μην υπάρχει η δυνατότητα το text view να παίρνει αρνητικές τιμές.

Εάν ο διαχειριστής κάνει κάποιες αλλαγές στα στοιχεία του σταθμού μπορεί να τα αποθηκεύσει πατώντας το Αποθήκευση αλλαγών. Στην Εικόνα 86 φαίνεται ο κώδικας της αποθήκευσης αλλαγών.

```
236         saveChanges.setOnClickListener(new View.OnClickListener() {
237             @Override
238             public void onClick(View v) {
239                 int available_seats = Integer.parseInt(diatheseis.getText().toString().trim());
240                 int seats = Integer.parseInt(EditTheseis.getText().toString().trim());
241                 String name = parkingname.getText().toString().trim();
242                 String adress = EditAdress.getText().toString().trim();
243                 String phone = EditTel.getText().toString().trim();
244                 double lati = Double.parseDouble(EditTextLat.getText().toString().trim());
245                 double lngi = Double.parseDouble(EditTextLng.getText().toString().trim());
246                 double price = Double.parseDouble(EditPrice.getText().toString().trim());
247
248                 DatabaseReference dbRef=FirebaseDatabase.getInstance().getReference();
249                 dbRef.child("privateParkings").child(name).child("diatheseis").setValue(available_seats);
250                 dbRef.child("privateParkings").child(name).child("lat").setValue(lati);
251                 dbRef.child("privateParkings").child(name).child("lng").setValue(lngi);
252                 dbRef.child("privateParkings").child(name).child("adress").setValue(adress);
253                 dbRef.child("privateParkings").child(name).child("theseis").setValue(seats);
254                 dbRef.child("privateParkings").child(name).child("tel").setValue(phone);
255                 dbRef.child("privateParkings").child(name).child("price").setValue(price);
256
257                 Toast.makeText(getApplicationContext(), text: "Οι αλλαγές αποθηκεύτηκαν ", Toast.LENGTH_SHORT).show();
258             }
259         });
260
261     }
```

Εικόνα 86: Αποθήκευση αλλαγών στοιχείων σταθμού

Όπως φαίνεται από την παραπάνω Εικόνα αρχικά δημιουργούνται μεταβλητές με τις τιμές των πεδίων. Για την αντικατάσταση των τιμών χρησιμοποιείται η setValue με παράμετρο την μεταβλητή του αντίστοιχου πεδίου. Η firebase έχει την δυνατότητα ελέγχου των παλαιών τιμών των πεδίων με τα νέα και κάνει αναβάθμιση της τιμής μόνο στα πεδία που έχει αλλάξει η τιμή τους.

3.9 Επίλογος

Στο κεφάλαιο αυτό παρουσιάστηκαν η βάση δεδομένων Firebase που χρησιμοποιείται στην εφαρμογή, η δομή του project στο Android Studio, τα αρχεία grandle και το AndroidManifest καθώς και ο κώδικας java. Συνολικά η εφαρμογή από πέντε κλάσεις και 14 Activities.

Κεφάλαιο 4ο: Συμπεράσματα και προτάσεις βελτίωσης

Ολοκληρώνοντας την ανάπτυξη της εφαρμογής, ένα από τα πιο συμαντικά συμπεράσματα είναι ότι για την κατασκευή μιας σωστά δομημένης εφαρμογής χρειάζεται αρκετή προγραμματιστική εμπειρία και γνώση των τεχνολογιών που χρησιμοποιούνται. Η επιλογή των κατάλληλων τεχνολογιών βοηθάει σημαντικά στην εκάστοτε ανάπτυξη της εφαρμογής.

Μελλοντικές βελτιώσεις για την εφαρμογή θα μπορούσε να είναι η προσθήκη περισσότερων ρυθμίσεων στο προφίλ του χρήστη. Ένα παράδειγμα είναι η προσθήκη σκούρο θέματος στην εφαρμογή (dark mode) ώστε να είναι πιο ξεκούραστη η χρήση της εφαρμογής κατά την διάρκεια της νύχτας. Επίσης μια βελτίωση είναι η δημιουργία πραγματικών συναλλαγών στην εφαρμογή μεταξύ των χρηστών και σταθμών στάθμευσης και όχι εικονική όπως είναι τώρα. Ο χρήστης θα μπορεί μέσω της χρεωστικής του κάρτας να προσθέτει χρήματα στην εφαρμογή και από εκεί να πραγματοποιεί συναλλαγές με τον κάθε σταθμό στάθμευσης.

Μια από τις σημαντικότερες βελτιώσεις στην εφαρμογή είναι η ανακατασκευή του κώδικα με βάση κάποιο design pattern όπως για παράδειγμα το MVP (Model View Presenter). Η ανακατασκευή του κώδικα με βάση κάποιο design pattern ο κώδικας θα οργανωθεί με κάποιους κανόνες και έτσι θα είναι πιο ευανάγνωστος προς τον προγραμματιστή που έχει αναπτύξει την εφαρμογή αλλά και σε κάποιον τρίτο, καθώς και η συντήρηση του κώδικα γίνεται πιο εύκολα και πιο γρήγορα.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] Paul Deitel , Harvey Deitel, Abbey Deitel, *Android – How to Program*. SUDBARY, MA: Deitel & Associates Inc, 2014.

[2] Walter Savitch, Kenrick Monk, *JAVA 7th Edition*. San Diego, CA: Pearson, 2014.

[3] Ε. Κεραμόπουλος, "Προηγμένα Θέματα Αλληλεπίδρασης Ανθρώπου - Μηχανής", Διεθνές Πανεπιστήμιο της Ελλάδας, Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων.

[4] Android Developers Documentation, "Android Architecture" [Online]. Available: https://source.android.com/docs/core/architecture.

[5] Android Developers Documentation, "Activity lifecycle" [Online] Available: https://developer.android.com/guide/components/activities/activity-lifecycle.

[6] Android Developers Documentation, "Activity state changes" [Online] Available: https://developer.android.com/guide/components/activities/state-changes.

[7] Android Developers Documentation, "Services overview" [Online] Available: https://developer.android.com/guide/components/services.

[8] Android Developers Documentation, "Content provider basics" [Online] Available: https://developer.android.com/guide/topics/providers/content-provider-basics.

[9] Android Developers Documentation, "Broadcast overview" [Online] Available: https://developer.android.com/guide/components/broadcasts.

[10] Android Developers Documentation, "Platform Arcitecture" [Online] Available: https://developer.android.com/guide/platform.

[11] Android Developers Documentation, "App Manifest overview" [Online] Available: https://developer.android.com/guide/topics/manifest/manifest-intro.

[12] Android Developers Documentation, "Manifest permission" [Online] Available: https://developer.android.com/reference/android/Manifest.permission.

[13] Firebase Documentation, "Firebase Realtime Database" [Online] Available: https://firebase.google.com/docs/database.

[14] Firebase Documentation, "Cloud Firestore" [Online] Available: https://firebase.google.com/docs/firestore.

[15] Firebase Documentation, "Authentication with Firebase Using Password-Based Accounts on Android" [Online] Available: https://firebase.google.com/docs/auth/android/password-auth.

[16] Firebase Documentation, "Authentication with Google on Android" [Online] Available: https://firebase.google.com/docs/auth/android/google-signin.

[17] Firebase Documentation, "Authentication using Github on Android" [Online] Available: https://firebase.google.com/docs/auth/android/github-auth.

[18] RecyclerView, "Simple Explanation of RecyclerView in Android" [Online] Available: https://www.youtube.com/watch?v=uh6lKnfp5hY.

[19] Google Maps Platform, “Google Maps Platform Billing Account” [Online] Available: [19] Google Maps Platform, “Google Maps Platform APIs” [Online] Available: https://developers.google.com/maps/billing-and-pricing/billing?hl=en_US#maps-product.

[20] Google Maps Platform, “Google Maps Platform APIs” [Online] Available: https://console.cloud.google.com/google/maps-apis/api-list.

[21] Google Maps Platform, “Google Maps Platform Credentials” [Online] Available: https://console.cloud.google.com/google/maps-apis/credentials?project=ptixiaki-351613.