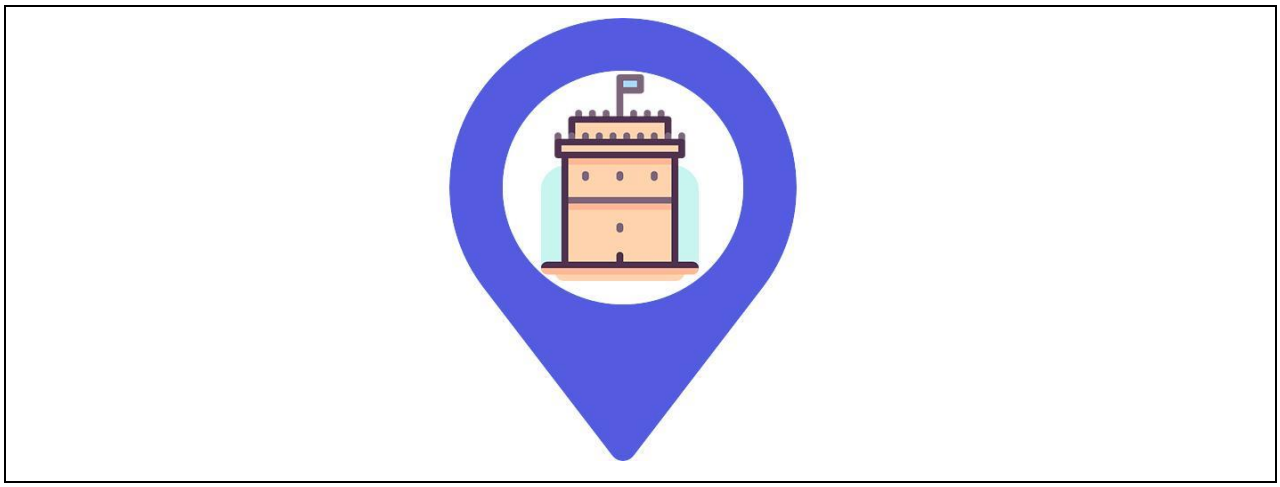
ΔΙΕΘΝΕΣ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΤΗΣ ΕΛΛΑΔΟΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

«Εφαρμογή parking με τη χρήση react native»



Του φοιτητή:
Νίκα Θωμά
Αρ. Μητρώου:154512

Επιβλέπων
Ονοματεπώνυμο: Αντώνης
Σιδηρόπουλος
Βαθμίδα:Επίκουρος Καθηγητής

Ημερομηνία 4/8/2021

Τίτλος Δ.Ε. **Εφαρμογή parking με τη χρήση react native**

Κωδικός Δ.Ε. **21110**

Ονοματεπώνυμο φοιτητή/τών **ΝΙΚΑ ΘΩΜΑΣ**

Ονοματεπώνυμο εισηγητή **ΑΝΤΩΝΗΣ ΣΙΔΗΡΟΠΟΥΛΟΣ**

Ημερομηνία ανάληψης Δ.Ε. **16-02-2021**

Ημερομηνία περάτωσης Δ.Ε. **14-09-2021**

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

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

Πρόλογος

Το πρόβλημα του παρκινγκ αποτελεί μείζων ζήτημα για πολλές πόλεις τα τελευταία χρόνια. Σε ώρες αιχμής η εύρεση παρκινγκ μπορεί να φτάσει μέχρι και τριάντα λεπτά. Πιο συγκεκριμένα για την πόλη της Θεσσαλονίκης όπου η έλλειψη συγκοινωνιών οδηγεί πολλούς εργαζόμενους στην καθημερινή χρήση του αυτοκινήτου, με αποτέλεσμα την δημιουργία διαρκής κίνησης ιδιαίτερα τις μεσημεριανές ώρες. Η εφαρμογή ThessParking στοχεύει στη βελτίωση της συγκοινωνιακής συμφόρησης προτείνοντας στον χρήστη το κατάλληλο για αυτόν παρκινγκ. Ο χρήστης έχει την δυνατότητα να δημιουργήσει λογαριασμό στην εφαρμογή αφού πρώτα συμφωνήσει για την επεξεργασία των προσωπικών δεδομένων του, που θα αναλυθούν παρακάτω, να βαθμολογεί, να αποθηκεύει τα αγαπημένα του παρκινγκ και να μπορεί να κάνει κράτηση. Για κάθε χώρο στάθμευσης παρουσιάζονται αναλυτικά στοιχεία με τιμές, τοποθεσία βαθμολογία κλπ. δίνοντας πλήρη γνώση στο χρήστη για το εκάστοτε παρκινγκ. Για τους ιδιοκτήτες παρκινγκ έχει δημιουργηθεί μία ξεχωριστή καρτέλα όπου θα έχουν πρόσβαση μετά από διασταύρωση στοιχείων. Εκεί θα μπορούν να δουν τις προτάσεις που τους παρουσιάζονται για διαφήμιση του χώρου στάθμευσης τους, μέσα στην εφαρμογή με τρεις τρόπους. Επιπρόσθετα στην ενότητα στατιστικά μπορούν να δούνε με γραφήματα την επισκεψιμότητα για τα παρκινγκ τους καθώς και πόσοι χρήστες τα έχουν προσθέσει στα αγαπημένα τους.

Περίληψη

Το αντικείμενο της πτυχιακής εργασίας είναι η δημιουργία μιας εφαρμογής παρκινγκ για κινητά τηλέφωνα. Η υλοποίηση της εφαρμογής έγινε με την χρήση της react native που αποτελεί μια βιβλιοθήκη της JavaScript. Για την δημιουργία της βάσης χρησιμοποιήθηκε το firebase της Google τόσο για το authentication όσο και για την αποθήκευση των δεδομένων. Η εφαρμογή αποσκοπεί στο να δίνει την δυνατότητα στο χρήστη να βρίσκει το κατάλληλο για αυτόν παρκινγκ για την πόλη της Θεσσαλονίκης ανάλογα με την τοποθεσία του χρήστη καθώς και την τιμή του εκάστοτε παρκινγκ. Εκτός από τους απλούς χρήστες η εφαρμογή προσφέρει κάποιες δυνατότητες και στους ιδιοκτήτες όπως προτάσεις για διαφήμιση ή στατιστικά στοιχεία για το παρκινγκ τους.

Αρχικά θα γίνει μία μικρή αναφορά στο πρόβλημα του παρκινγκ που αποτελεί μείζων ζήτημα για πολλές πόλεις. Στη συνέχεια θα αναλυθούν οι τεχνολογίες που χρησιμοποιήθηκαν και ο λόγος που επιλέχθηκαν αυτές. Επίσης θα αναφερθεί το χρονολογικό πλαίσιο της υλοποίησης της εφαρμογής μέχρι το τελικό αποτέλεσμα. Έπειτα θα αναλυθεί η δομή της εφαρμογής για κάθε οθόνη ποια components χρησιμοποιήθηκαν, που υπάρχουν class και function components. Τέλος θα αναφερθούν μελλοντικές προτάσεις για βελτιώσεις, αλλαγές και προσθήκες που μπορούν να γίνουν, μαζί με συμπεράσματα για την εφαρμογή και για το πρόβλημα του παρκινγκ γενικότερα.

«React native Parking app»

(στην αγγλική γλώσσα)

«Thomas Nika»

(στην αγγλική γλώσσα)

Abstract

The subject of this final thesis is the creation of a Parking app for mobile phones. The implication of the app was made in react native who constitutes a JavaScript library. In order to create the database the app was used by firebase from Google so for authentication as for the data storage. The app aims to give the ability to the user so that he can find the best for him parking for the city of Thessaloniki which depends on the user location and the price of the parking. Except from normal users, the app has some possibilities for the parking owners. The owners can advertise their parking in the app and they can see statistics for their parking.

At first there will be one small reference to the Parking problem that is a major issue for many cities. Subsequently will analyze the technologies that have been used and the reason why these technologies used. Also it will be told the time frame of the creation of the app until the final result. Afterwards the structure of the application will be analyzed for each screen which components were used and which of them are class or function components. At the end there will be future suggestions for improvement changes and ads that can happen and as well inferences for the app and the parking problem in general.

Ευχαριστίες

Θεωρώ υποχρέωσή μου να ευχαριστήσω τον επιβλέποντα καθηγητή κ. Αντώνη Σιδηρόπουλο για την πολύτιμη καθοδήγησή του. Επιπρόσθετα ένα μεγάλο ευχαριστώ στους γονείς και την οικογένεια μου που με στήριξαν όλα τα χρόνια της φοίτησης στο Δ.Ι.Π.Α.Ε.

Περιεχόμενα

Κατάλογος Σχημάτων

Κατάλογος Εικόνων

Συντομογραφίες

Π.Ε.	Πτυχιακή Εργασία
ΔΙΠΑΕ	Διεθνές Πανεπιστήμιο Ελλάδος
Π.Ε.	Πτυχιακή Εργασία
RN	React Native
JS	Javascript

JSX	JavaScript XML
XML	extensible markup language
Html	Hypertext Markup Language
OSM	Open Street Maps
VS Code	Visual Studio Code

Κεφάλαιο 1ο: Τεχνολογίες που χρησιμοποιήθηκαν

1.1 Εισαγωγή

Στο κεφάλαιο αυτό θα γίνει ανάλυση των τεχνολογιών που εφαρμόστηκαν στη εφαρμογή, καθώς και γιατί προτιμήθηκαν από άλλες. Θα γίνει αναφορά τόσο στο frontend όσο και στο backend, θα αναλυθεί εκτενώς η γλώσσα προγραμματισμού που χρησιμοποιήθηκε καθώς και τα προγράμματα και οι λειτουργίες που βοηθούν στον ευκολότερο προγραμματισμό της εφαρμογής

1.2 React Native

1.2.1 Λίγα λόγια για την React Native και τις δυνατότητές της

Η κύρια γλώσσα προγραμματισμού που χρησιμοποιήθηκε για την υλοποίηση της εφαρμογής ήταν η react native[3]. Η React Native (γνωστή και ως RN) είναι δημοφιλής Javascript-based για εφαρμογές κινητών. Το framework επιτρέπει στους χρήστες να δημιουργούν εφαρμογές για κινητές συσκευές, natively-rendered για iOS και Android. Εκτός από την δυνατότητα που δίνει να χρησιμοποιείς ουσιαστικά μόνο μία γλώσσα προγραμματισμού, έχει την ιδιότητα σχεδόν ο ίδιος κώδικας που χρησιμοποιείται για android να χρησιμοποιείται και για ios, χωρίς να χρειάζεται ο προγραμματιστής να φτιάχνει ξεχωριστά προγράμματα για κάθε πλατφόρμα. Η RN κυκλοφόρησε για πρώτη φορά από το Facebook ως open-source project το 2015. Σε λίγα μόλις χρόνια, έγινε μία από τις κορυφαίες λύσεις που χρησιμοποιήθηκαν για την ανάπτυξη εφαρμογών. Πολλές κορυφαίες εφαρμογές παγκοσμίως χρησιμοποιούν για την ανάπτυξή των εφαρμογών τους RN, όπως το Instagram, το Facebook και το Skype[2]. Η RN αυξάνεται σε δημοτικότητα ως μία πολύ καλή λύση για την δημιουργία εφαρμογών σε κινητά με μικρότερο φόρτο στον προϋπολογισμό της επιχείρησης. Μεταξύ των κορυφαίων 500 εφαρμογών στις ΗΠΑ, το 14,85% των εγκατεστημένων εφαρμογών είναι κατασκευασμένες με RN. Στην πραγματικότητα είναι η 3η δημοφιλέστερη μετά από την Kotlin και Android. Βρίσκεται υπό συνεχή ανάπτυξη τόσο από το facebook όσο και από την τεράστια κοινότητα που διαθέτει, που εργάζονται συνεχώς για την βελτίωση της. Εάν δεν υπάρχει μία λύση εκείνη την στιγμή, πολύ πιθανόν μέσα σε λίγο καιρό η κατάσταση να είναι διαφορετική. Επίσης επειδή οι διαφορές με την React δεν είναι πολλές, μπορεί εύκολα να μεταφερθεί σε web view.

Υπάρχουν πολλοί λόγοι πίσω από την επιτυχία της react:

- Χρησιμοποιώντας RN, οι εταιρείες μπορούν να δημιουργήσουν κώδικα μόνο μία φορά και να τον χρησιμοποιήσουν για να τροφοδοτήσουν τις εφαρμογές τους iOS και Android. Αυτό μεταφράζεται σε τεράστια εξοικονόμηση χρόνου και πόρων.
- Δημιουργήθηκε με βάση την React - μια βιβλιοθήκη JavaScript, η οποία ήταν ήδη εξαιρετικά δημοφιλής όταν κυκλοφόρησε το framework για κινητά.
- Το framework βοήθησε front end developers, που πριν μπορούσαν να δουλέψουν μόνο με web-based technologies
- Παρέχει τη δυνατότητα της γρήγορης ανανέωσης που επιτρέπει στους προγραμματιστές να εκτελούν την εφαρμογή ενώ την ενημερώνουν σε νέες εκδόσεις και τροποποιούν το περιβάλλον του χρήστη, οι αλλαγές είναι ορατές αμέσως. Αυτό οδηγεί σε δύο σημαντικά οφέλη: εξοικονόμηση χρόνου, καθώς οι προγραμματιστές εξοικονομούν χρόνο στη σύνταξη και αυξημένη παραγωγικότητα.

Λαμβάνοντας υπόψη τον ρυθμό με τον οποίο το framework ανέλαβε την αγορά και την απλή προσέγγισή του για την επίλυση προβλημάτων ανάπτυξης, το μέλλον της RN φαίνεται λαμπρό.

Παρόλο που έχει μερικά μειονεκτήματα, τα οποία θα αναλυθούν παρακάτω, η ταχύτητα και η ευκολία ανάπτυξης του τα αντισταθμίζει.

1.2.2 Πως δουλεύει;

Η RN γράφεται με ένα μείγμα JavaScript και JXL, έναν ειδικό κώδικα σήμανσης που μοιάζει με XML. Το framework έχει τη δυνατότητα να επικοινωνεί και με τα δύο πεδία που βασίζονται σε JavaScript. Υπάρχουν δύο σημαντικά threads που εκτελούνται σε κάθε εφαρμογή React Native. Ένα από αυτά είναι το κύριο thread, το οποίο εκτελείται σε κάθε τυπική native εφαρμογή. Χειρίζεται την εμφάνιση των στοιχείων της διεπαφής χρήστη και επεξεργάζεται τα gestures του χρήστη. Το άλλο thread είναι συγκεκριμένο για την React Native. Ο στόχος του είναι να εκτελέσει τον κώδικα JavaScript σε ξεχωριστή μηχανή JavaScript. Η JavaScript ασχολείται με την επιχειρησιακή λογική της εφαρμογής. Καθορίζει επίσης τη δομή και τις λειτουργίες του user interface. Αυτά τα δύο threads δεν επικοινωνούν ποτέ άμεσα και ποτέ δεν μπλοκάρουν το ένα το άλλο. Μεταξύ αυτών των δύο threads βρίσκεται η λεγόμενη γέφυρα, η οποία είναι ο πυρήνας της React Native. Η γέφυρα έχει τρία σημαντικά χαρακτηριστικά. Επιτρέπει την ασύγχρονη επικοινωνία μεταξύ των threads. Αυτό εξασφαλίζει ότι δεν μπλοκάρουν ποτέ το ένα το άλλο. Μεταφέρει μηνύματα από το ένα νήμα στο άλλο με βελτιστοποιημένο τρόπο και τα δύο νήματα δεν μοιράζονται ή λειτουργούν ποτέ με τα ίδια δεδομένα. Αντ 'αυτού, ανταλλάσσουν σειριακά μηνύματα.

Ακολουθεί μια απεικόνιση της έννοιας της γέφυρας:

Σχήμα 1.1 Η έννοια της γέφυρας

Αυτό σημαίνει ότι εάν κάποιος χρήστης έχει ήδη μια native εφαρμογή iOS ή Android - μπορεί ακόμα να χρησιμοποιήσει τα στοιχεία της ή να μεταβεί στην ανάπτυξη RN. Η διαφορά μεταξύ της RN και άλλων λύσεων ανάπτυξης πολλαπλών πλατφόρμων (για παράδειγμα, Cordova και PhoneGap) είναι ότι η RN δεν παρέχει WebViews στον κώδικά της. Λειτουργεί με πραγματικές, native views και components. Αυτός είναι ένας από τους λόγους για τη θεαματική επιτυχία της.

1.2.3 Κίνδυνοι και μειονεκτήματα

Ακολουθούν τα τρία κορυφαία πιθανά μειονεκτήματα που πρέπει να γνωρίζει κάθε χρήστης πριν αποφασίσει να αναπτύξει μια εφαρμογή σε RN.

- Ορισμένα modules είτε αφήνουν περιθώρια βελτίωσης είτε λείπουν εντελώς. Αυτό σημαίνει ότι ίσως χρειαστεί να τρέξει ο χρήστης τρεις ξεχωριστές βάσεις κώδικα (για RN, iOS και Android) αντί για μία, αλλά δεν είναι συνηθισμένο φαινόμενο.

- Παρόλο που μπορεί να είναι έκπληξη αφού χρησιμοποιείται από κορυφαίες εταιρείες τεχνολογίας - είναι ακόμα σε beta φάση. Οι προγραμματιστές ενδέχεται να αντιμετωπίσουν διάφορα προβλήματα με τη συμβατότητα πακέτων ή τα εργαλεία εντοπισμού σφαλμάτων. Εάν οι προγραμματιστές δεν είναι αρκετά καλοί, αυτό μπορεί να επηρεάσει αρνητικά την ανάπτυξή της εφαρμογής καθώς θα αφιερώνουν χρόνο σε μακροχρόνια αντιμετώπιση προβλημάτων.
- Τις περισσότερες φορές, η react θα λειτουργήσει πολύ καλά, ακόμη και αν η εφαρμογή είναι περίπλοκη. Άλλωστε, εταιρείες όπως το Facebook και το Skype χρησιμοποιούν το framework με συνέπεια για πολλά χρόνια. Ορισμένες εταιρείες αποφάσισαν να εγκαταλείψουν τη χρήση του RN. Η Airbnb, για παράδειγμα, αποφάσισε να χρησιμοποιήσει το framework για την εφαρμογή της όταν ήταν απλώς μια αναδυόμενη νεοσύστατη εταιρεία. Με την πάροδο του χρόνου, ωστόσο, αποδείχθηκε ακατάλληλη για τα σχέδια ανάπτυξης της εταιρείας και η Airbnb κατέφυγε στην ανάπτυξη δύο native εφαρμογών. Με τις τρέχουσες εξελίξεις στη RN και με τις σωστές επιλογές αρχιτεκτονικής λογισμικού, τα προβλήματα επεκτασιμότητας μπορούν εύκολα να αποφευχθούν.

1.2.4 Άλλες λύσεις

Η άλλη λύση για την υλοποίηση της εφαρμογής ήταν το android. Προφανώς η πρώτη διαφορά έγκειται στο γεγονός ότι στη react με σχεδόν ίδιο κώδικα μπορείς να τρέξεις την εφαρμογή σε ios και android με ελάχιστες παραλλαγές. Επίσης η λειτουργία της react είναι πολύ πιο εύκολη καθώς με βασικές γνώσεις JavaScript μπορείς να φτιάξεις πολύπλοκες εφαρμογές απλά εισάγοντας τις κατάλληλες βιβλιοθήκες που θα βοηθήσουν στην κάθε περίπτωση και υπάρχουν χιλιάδες βιβλιοθήκες σχεδόν για τα πάντα. Για την υλοποίηση στο android χρησιμοποιούνται kotlin και java και για ios Swift ή Objective-C ενώ με την react μόνο JSX. Προφανώς όλο αυτό επηρεάζει και τους διαθέσιμους πόρους που θα χρησιμοποιηθούν, με την react να χρησιμοποιεί λιγότερους.

Συνοψίζοντας, οι εφαρμογές με RN είναι εύκολο να γραφτούν εξοικονομώντας χρόνο για τους προγραμματιστές και μειώνοντας το κόστος για τους διαχειριστές των έργων. Μειώνουν το κόστος ανάπτυξης και συντήρησης, επειδή δεν χρειάζεται να ασχοληθεί ο προγραμματιστής με 2 ξεχωριστές γλώσσες για ios και Android. Τέλος δεν είναι απαγορευτικό η χρήση κώδικα Java ή Swift όπου χρειάζεται.

1.3 Class based components vs Function based components

Στον κόσμο της React, υπάρχουν δύο τρόποι γραφής. Ο ένας χρησιμοποιεί συναρτήσεις και το άλλος χρησιμοποιεί κλάσεις. Στην εφαρμογή χρησιμοποιήθηκαν κατά κύριο λόγο Class based components. Στη συνέχεια αναλύονται οι δύο τρόποι γραφής αλλά και γιατί επιλέχθηκαν τα class based components.

1.3.1 Rendering JSX

Πρώτα απ 'όλα, η σαφής διαφορά είναι η σύνταξη. Ακριβώς όπως στα ονόματά τους, ένα function component είναι μία απλή συνάρτηση JavaScript που επιστρέφει JSX. Ένα στοιχείο κλάσης είναι μια κλάση JavaScript που επεκτείνει το “React.Component” το οποίο έχει μια μέθοδο render. Ακολουθεί παράδειγμα κώδικα.

```
import React from "react";
```

```
const FunctionalComponent = () => {
 return <h1>Hello, world</h1>;
};
```

Σύνταξη functional component

Όπως φαίνεται, ένα functional component, είναι μία συνάρτηση που επιστρέφει JSX. Από την άλλη πλευρά, όταν ορίζεται ένα class components, πρέπει να δημιουργηθεί μια κλάση που επεκτείνει το "React.Component".

```
import React, { Component } from "react";

class ClassComponent extends Component {
 render() {
   return <h1>Hello, world</h1>;
 }
```

Σύνταξη class component

1.3.2 Διαχείριση μεταβλητών

Η μεταφορά μεταβλητών μπορεί να είναι περίπλοκη και διαφέρει στους 2 τρόπους σύνταξης. Δηλώνεται αρχικά η συνάρτηση και στην συνέχεια ανάμεσα στις παρενθέσεις περνάει η μεταβλητή που θέλει να χρησιμοποιήσει ο χρήστης. Για την αναφορά των μεταβλητών σε κλάσεις, χρησιμοποιείται το props (props.name). Το οποίο δηλώνεται μέσα στο render και μετά απλά ο χρήστης με την χρήση {} και το όνομα της, μπορεί να χρησιμοποιεί την μεταβλητή όπου θέλει.

1.3.3 setState-useState

Η χρήση των state στην react είναι κάτι που σίγουρα υπάρχει σε κάθε project. Ουσιαστικά, όταν ο χρήστης θέλει να αλλάξει το περιεχόμενο μιας μεταβλητής και αυτό να ανανεωθεί αυτόματα χρησιμοποιεί το state. Η διαχείριση των state ήταν εφικτή μόνο από class components μέχρι την έκδοση της react 16.8, όπου και έγινε η εισαγωγή του React hook useState και των *hooks* γενικότερα που αποτελούν το σημαντικότερο εργαλείο των functional components και ένας από τους λόγους που κάποιοι χρήστες τα προτιμούν από τα class. Το useState Hook παίρνει μία μεταβλητή ως initial state, αυτό μπορεί να είναι αριθμός, string, αντικείμενο ή οποιοσδήποτε τύπος μεταβλητής υποστηρίζει η JavaScript. Στην αριστερή πλευρά του useState υπάρχει η αρχική μεταβλητή και η συνάρτηση που την κάνει update (π χ [count, setCount] = React.useState(0);).

Το class based component χειρίζεται λίγο διαφορετικά την κατάσταση, αλλά με την ίδια φιλοσοφία. Αρχικά πρέπει να γίνει ορισμός του React.component: Ο constructor για ένα React component καλείται πριν τοποθετηθεί. Κατά την εφαρμογή του constructor πρέπει ο χρήστης να καλέσει super (props), πριν από οποιαδήποτε άλλη κλήση, αλλιώς μπορεί να οδηγηθεί σε σφάλματα ο κώδικας γιατί όλες οι μεταβλητές κατάστασης θα είναι απροσδιόριστες. Μέσα στον constructor δημιουργείται το αντικείμενο με την αρχική τιμή, μέσα στο JSX με την χρήση του this.state έχει πρόσβαση στην μεταβλητή και με την χρήση του this.setState({}) μπορεί να αλλάξει τα περιεχόμενά της[14,15].

1.3.4 Σύνοψη

Τα functional components γράφονται συντομότερα και απλούστερα που βοηθάει στην εύκολη ανάπτυξη αλλά χρειάζεται μεγάλος χρόνος εξοικείωσης και αποτελούν ένα νέο εργαλείο με αποτέλεσμα πολλές βιβλιοθήκες να είναι βασισμένες σε class components. Τα class components

μπορούν να μπερδέψουν εύκολα τον χρήστη ειδικά όταν υπάρχει μεγάλος αριθμός μέσα στο πρόγραμμα, παρόλα αυτά είναι πιο εύκολα στην κατανόηση και με σωστό προγραμματισμό του χρήστη αποτελούν ιδανική λύση.

1.4 Firebase

Για το backend χρησιμοποιήθηκε το firebase από την Google που δίνει πολλές δυνατότητες στο χρήστη. Αρχικά για το authentication δίνονται πολλές επιλογές για το πως θα κάνει σύνδεση ο χρήστης. Η ενημέρωση των στοιχείων είναι ταχύτατη και ασφαλέστατη χωρίς να υπάρχει κίνδυνος υποκλοπής στοιχείων.

Το Mobile Backend as a Service (MBaas) είναι ένα μέσο που προσφέρει έναν τρόπο σύνδεσης του ιστού και των εφαρμογών με το cloud και τα APIs του backend. Κοινώς γνωστό ως Backend as a Service (BaaS), το MBaaS προσφέρει δυνατότητες όπως αποστολή push notifications, ενσωμάτωση του cloud storage, διαχείριση χρηστών και κοινωνικά δίκτυα. Το firebase μπορεί να χρησιμοποιηθεί για τα ακόλουθα:

- Να διαχειρίζεται όλα τα δεδομένα σε πραγματικό χρόνο στη βάση δεδομένων. Έτσι, η ανταλλαγή δεδομένων από και προς τη βάση δεδομένων είναι εύκολη και γρήγορη. Επομένως, αναπτύσσονται εφαρμογές για κινητά, όπως live streaming, μηνύματα συνομιλίας κ.λπ., μπορούν οι χρήστες να χρησιμοποιούν το Firebase.
- Το Firebase επιτρέπει τον συγχρονισμό των δεδομένων σε πραγματικό χρόνο σε όλες τις συσκευές- Android, iOS και web χωρίς ανανέωση της οθόνης
- Το Firebase προσφέρει άλλες δυνατότητες όπως Google Ads, AdMob, DoubleClick, Play Store, Data Studio, BigQuery και Slack, για να κάνει την ανάπτυξη της εφαρμογής πιο εύκολη και αποτελεσματική

Τα πλεονεκτήματά του είναι:

- Δημιουργία εφαρμογής χωρίς backend server, και αυτό συνεπάγεται με το ότι δεν χρειάζονται παραπάνω χρήματα
- Τα δεδομένα συγχρονίζονται σε πραγματικό χρόνο και εμφανίζονται γρήγορα
- Πιο γρήγορο από οποιαδήποτε backend services αλλά και από βάσεις sql

Για το data storage δίνει 2 επιλογές το real-time database που αποτελεί την πιο απλή έκδοση βάσης δεδομένων και λιγότερο χρησιμοποιημένο τα τελευταία χρόνια και η άλλη επιλογή είναι το firestore database.

1.4.1 Firestore

Το firestore αποτελεί μία γρήγορη και αξιόπιστη λύση (π.χ. αν προστεθεί καινούργιο παρκινγκ αμέσως θα εμφανιστεί στην εφαρμογή ή εάν αλλάξει κάποιο rating ο χρήστης αμέσως θα περαστεί στην βάση με σχεδόν μηδαμινή χρονική καθυστέρηση). Επιπρόσθετα έχει την δυνατότητα για offline support και οι περισσότερες δυνατότητες είναι δωρεάν χωρίς κάποια οικονομική επιβάρυνση. Τα δεδομένα μπορούν να δημιουργηθούν χειροκίνητα ή ακόμα και με την χρήση της JavaScript (σε πολλές περιπτώσεις όταν έπρεπε να περάσουν πολλές εγγραφές μαζί απλά με το πάτημα του npm start, σε ελάχιστο χρόνο εισέρχονταν οι εγγραφές). Κατά κύριο λόγο ο χρήστης έχει πρόσβαση στο firestore με async functions. Το Cloud Firestore μπορεί να κλιμακωθεί καλύτερα από τη βάση δεδομένων σε πραγματικό χρόνο, δηλαδή η αναζήτηση θα παραμείνει γρήγορη, ανεξάρτητα από το πόσο μεγάλο μπορεί να είναι το σύνολο δεδομένων. Μπορεί κάποιες βάσεις δεδομένων να έχουν πιο εύχρηστο περιβάλλον όπως το mongoDB αλλά το firestore ταιριάζει ιδανικά με την react αφού υπάρχουν επιπλέον συγκεκριμένες βιβλιοθήκες για το firebase (@react-native-firebase/firestore)[11].

1.4.2 Admob

Το admob αποτελεί ακόμα ένα πολύ χρήσιμο εργαλείο που υπάρχει στο firebase και χρησιμοποιήθηκε στην εφαρμογή. Με το *admob* ο χρήστης μπορεί να προβάλει διάφορους τύπους διαφημίσεων μέσα στην εφαρμογή και να διαφημίσει τους πελάτες του. Οι δυνατότητες διαφήμισης κυμαίνονται σε 3 άξονες.

Στα *banner ads* καταλαμβάνουν ένα σημείο στη διάταξη μιας εφαρμογής το οποίο συνήθως είναι στο πάνω ή κάτω μέρος. Παραμένουν στη οθόνη ενώ οι χρήστες αλληλοεπιδρούν με την εφαρμογή και μπορούν να ανανεώνονται αυτόματα μετά από ένα χρονικό διάστημα. Τα κινητά τηλέφωνα έχουν περιορισμένο μέγεθος οθόνης άρα η τοποθέτηση του *banner ad* πρέπει να γίνει προσεκτικά. Για να αποφεύγονται τυχαία κλικ στην διαφήμιση οι διαφημίσεις δεν πρέπει να τοποθετούνται δίπλα σε κουμπιά όπως επόμενο ή προηγούμενο.

Η άλλη κατηγορία διαφημίσεων είναι οι *Interstitial ads*. Είναι διαφημίσεις πλήρους οθόνης που ο χρήστης μπορεί να τις κλείσει πατώντας στο Χ που βρίσκεται πάνω δεξιά ή μπορεί να επιλέξει να πατήσει πάνω στη διαφήμιση και να ακολουθήσει τον σύνδεσμο. Συνήθως εμφανίζονται κατά την διάρκεια αλληλεπίδρασης του χρήστη με την εφαρμογή. Οι *Interstitial ads* θα πρέπει να τοποθετούνται προσεκτικά καθώς σε κύρια σημεία π χ όταν κάνουν log in οι χρήστες ίσως δημιουργήσει εκνευρισμό του χρήστη ή ακόμα και σε data loss.

Η τελευταία κατηγορία είναι οι *rewarded Ads* που ανταμείβουν τους χρήστες εντός της εφαρμογής εάν παρακολουθήσουν κάποιο βίντεο για λίγα δευτερόλεπτα. Οι διαφημίσεις μπορούν να προβληθούν μόνο εάν ο προγραμματιστής επιλέξει να εμφανιστεί η διαφήμιση στο συγκεκριμένο σημείο, δίνοντάς του έτσι το απόλυτο έλεγχο. Για παράδειγμα: Ένας χρήστης που παίζει ένα παιχνίδι και δεν έχει άλλες ζωές. Ο χρήστης ερωτάται εάν θέλει να ανακτήσει αμέσως ζωές του παρακολουθώντας μια διαφήμιση βίντεο. Εναλλακτικά, μπορεί να ξοδέψει χρήματα ή να περιμένει ένα συγκεκριμένο χρονικό διάστημα για να φτάσει στο ίδιο αποτέλεσμα. Τέλος οι διαφημίσεις αυτές μετά την ολοκλήρωση του βίντεο εάν το παρακολούθησε όλο ο χρήστης, μπορούν να προτείνουν την εγκατάσταση ενός προγράμματος ή να ακολουθήσει κάποιο σύνδεσμο ο χρήστης που θα τον οδηγήσει σε άλλη σελίδα.

Η άλλη επιλογή για την δημιουργία διαφημίσεων ήταν οι facebook διαφημίσεις. Προτιμήθηκαν οι *admob* διαφημίσεις καθώς συνδέονται άμεσα με το firebase το οποίο χρησιμοποιεί και η εφαρμογή. Επίσης γιατί είναι προϊόν της Google έχει δυνατότητα πρόσβασης και εμφανίζει διαφημίσεις υψηλής ποιότητας και ο προγραμματιστής μπορεί να προσαρμόσει την διαφήμιση όπως αυτός θέλει. Επιπλέον υπάρχουν πολλές δυνατότητες και στατιστικά για το πόσοι π.χ. είδαν την διαφήμιση, χρήματα που κερδήθηκαν και πολλά ακόμη.

1.5 Visual Studio Code

Για την ανάπτυξη της εφαρμογής χρησιμοποιήθηκε το visual studio code, οι λόγοι που επιλέχθηκε η συγκεκριμένη πλατφόρμα ανάπτυξης λογισμικού αναλύονται στη συνέχεια. Το VS Code συνδυάζει την απλότητα ενός επεξεργαστή πηγαίου κώδικα με ισχυρά εργαλεία για προγραμματιστές, όπως η ολοκλήρωση και εντοπισμός σφαλμάτων κώδικα *IntelliSense*. Πρώτα απ 'όλα, είναι ένας editor που δεν εμποδίζει τον προγραμματιστή. Ο κύκλος edit-build-debug συνεπάγεται λιγότερο χρόνο στην ενασχόληση με το περιβάλλον και περισσότερο χρόνο για την εκτέλεση των ιδεών του χρήστη. Διατίθεται για macOS, Linux και Windows έτσι μπορεί ο προγραμματιστής να συνεχίσει να εργάζεται, ανεξαρτήτου πλατφόρμας.

1.5.1 Επεξεργασία, δημιουργία και εντοπισμός σφαλμάτων με ευκολία

Στην καρδιά του, το VS Code διαθέτει έναν επεξεργαστή πηγαίου κώδικα, ιδανικό για οποιονδήποτε χρήστη. Με υποστήριξη για εκατοντάδες γλώσσες, ο κώδικας VS Code βοηθά άμεσα τους χρήστες στο να είναι παραγωγικοί με syntax highlighting, bracket-matching, auto-indentation, box-selection, snippets,και άλλα. Οι συντομεύσεις πληκτρολογίου, η εύκολη προσαρμογή και οι αντιστοιχίσεις συντομεύσεων πληκτρολογίου που παρέχονται βοηθούν στην εύκολη πλοήγηση στον κώδικα και στην δημιουργία μεγάλων κομματιών του, απλά πατώντας μερικά πλήκτρα. Για σοβαρή κωδικοποίηση, συχνά είναι χρήσιμα τα εργαλεία με περισσότερη κατανόηση κώδικα από απλά κομμάτια κειμένου. Το VS Code περιλαμβάνει ενσωματωμένη υποστήριξη για την ολοκλήρωση του κώδικα *IntelliSense*, πλούσια κατανόηση και πλοήγηση σημασιολογικού κώδικα και αναδιαμόρφωση κώδικα. Και όταν η κωδικοποίηση γίνεται περίπλοκη, η πολυπλοκότητα μεταφέρεται και στο debugging. Ο εντοπισμός σφαλμάτων είναι συχνά το μόνο χαρακτηριστικό που λείπει περισσότερο από τους προγραμματιστές αυτό που μπορεί να οδηγήσει σε ώρες και ώρες ψαξίματος ενώ εν τέλει είναι κάτι πολύ απλό. Το VS περιλαμβάνει έναν διαδραστικό εντοπισμό σφαλμάτων, ώστε να μπορεί ο χρήστης να ελέγχει τον πηγαίο κώδικα, να εξετάζει τις μεταβλητές, να βλέπει τις στοίβες και να εκτελεί εντολές στο terminal. Ο κώδικας VS διαθέτει υποστήριξη για το *Git*, ώστε ο προγραμματιστή να εργάζεται με τον source control χωρίς να κλείνει τον editor, συμπεριλαμβανομένων των διαφορών προβολών.

1.5.2 Δυνατότητες προσαρμοστικότητας

Το VSCode παρέχει δυνατότητες προσαρμογής σύμφωνα με τις προτιμήσεις των χρηστών και δίνει την δυνατότητα εγκατάστασης οποιονδήποτε άλλων βιβλιοθηκών. Το VS Code είναι ένα έργο ανοιχτού κώδικα, ώστε να μπορεί να συνεισφέρει στην αναπτυσσόμενη και ζωντανή κοινότητα στο GitHub. Περιλαμβάνει εμπλουτισμένη ενσωματωμένη υποστήριξη για την ανάπτυξη του Node.js με JavaScript και TypeScript, που υποστηρίζονται από τις ίδιες τις τεχνολογίες που έχει το Visual Studio. Ο κώδικας VS περιλαμβάνει επίσης εξαιρετικά εργαλεία για τεχνολογίες web όπως JSX/React, HTML, CSS, SCSS, Less και JSON. Πρόκειται δηλαδή για ιδανική πλατφόρμα για την ανάπτυξη της RN.

1.5.3 Στιβαρή και επεκτάσιμη αρχιτεκτονική

Αρχιτεκτονικά, το VS Code συνδυάζει τις καλύτερες τεχνολογίες ιστού, native και language-specific. Χρησιμοποιώντας *Electron*, το VSCode συνδυάζει τεχνολογίες ιστού όπως JavaScript και Node.js με την ταχύτητα και την ευελιξία των native εφαρμογών. Ο κώδικας VS χρησιμοποιεί μια νεότερη, ταχύτερη έκδοση του ίδιου επεξεργαστή βιομηχανικής ισχύος που βασίζεται σε HTML και έχει τροφοδοτήσει τον επεξεργαστή cloud "Monaco", τα εργαλεία F12 του Internet Explorer και άλλα έργα. Επιπλέον, χρησιμοποιεί μια αρχιτεκτονική υπηρεσίας εργαλείων που του επιτρέπει να ενσωματωθεί με πολλές από τις ίδιες τεχνολογίες που τροφοδοτούν το Visual Studio, συμπεριλαμβανομένου του Roslyn για .NET, TypeScript, τον μηχανισμό εντοπισμού σφαλμάτων Visual Studio και πολλά άλλα. Περιλαμβάνει ένα δημόσιο μοντέλο επεκτασιμότητας που επιτρέπει στους προγραμματιστές να δημιουργούν και να χρησιμοποιούν επεκτάσεις και να προσαρμόζουν πλούσια την εμπειρία επεξεργασίας-δημιουργίας-εντοπισμού σφαλμάτων[13].

1.6 Google Maps

Για την δημιουργία των χαρτών που αποτελεί πολύ βασική λειτουργία της εφαρμογής χρησιμοποιήθηκαν οι χάρτες από την Google. Προσφέρει χάρτες δρόμων και σχεδιαστή διαδρομών για μεταφορές με τα πόδια, αυτοκίνητο, ποδήλατο (beta) ή μέσα μαζικής μεταφοράς. Περιλαμβάνει επίσης εντοπισμό των επιχειρήσεων που βρίσκονται σε πόλεις σε πολλές χώρες σε όλο τον κόσμο. Για να μπορέσει κάποιος να χρησιμοποιήσει αυτές τις δυνατότητες πρέπει να δημιουργήσει ένα λογαριασμό στο Google Cloud Platform και από εκεί να φτιάξει το project του. Το api που κάνει αιτήσεις ο χρήστης για να έχει πρόσβαση στην εφαρμογή είναι το Maps SDK for Android. Με την προσθήκη αυτού, τοποθετούνται οι χάρτες στην εφαρμογή μαζί με το Wear OS και δεδομένα των χαρτών. Η συγκεκριμένη υπηρεσία είναι δωρεάν όσα requests και αν κάνει ο χρήστης.

Μαζί με το Maps SDK for Android απαραίτητο για την δημιουργία διαδρομής στον προορισμό που έχει επιλέξει κάποιος χρήστης είναι το *directions api* [1] . Το *API Directions* είναι μια υπηρεσία ιστού που χρησιμοποιεί ένα αίτημα http για την επιστροφή οδηγιών μορφοποιημένων σε Json ή XML μεταξύ τοποθεσιών. Δίνει διάφορους τρόπους μεταφοράς όπως οδήγηση περπάτημα ή ποδηλασία. Το μόνο αρνητικό είναι ότι δεν είναι δωρεάν δηλαδή ανάλογα με τα αιτήματα που κάνει ο χρήστης πληρώνει και αντίστοιχο ποσό. Η τιμές κατά το 2021 είναι 5$ ανά 1000 αιτήματα άλλα Google δίνει σε όλους τους χρήστες 200$ κάθε μήνα για την χρήση των υπηρεσιών της. Οπότε για να πληρώσει πρακτικά κάποιος, ο αριθμός των αιτημάτων θα πρέπει να είναι αρκετά μεγάλος. Υπάρχουν πολλά διαφορετικά *directions api* προτιμήθηκε το api της Google γιατί είναι πολύ αξιόπιστο, γρήγορο και ευρέως χρησιμοποιούμενο.

Η άλλη επιλογή πέρα από τα Google maps ήταν τα openStreetmaps. Το OpenStreetMap (OSM) είναι ένας χάρτης ο οποίος αναπτύσσεται από μια κοινότητα εθελοντών που συνεισφέρουν και διατηρούν δεδομένα σχετικά με δρόμους, μονοπάτια, καφετέριες, σιδηροδρομικούς σταθμούς, και πολλά άλλα, σε όλο τον κόσμο. Οι συνεισφέροντες χρησιμοποιούν αεροφωτογραφίες, συσκευές GPS, και τοπικούς χάρτες χαμηλής τεχνολογίας για να είναι σίγουροι πως το OSM είναι ακριβής και ενημερωμένο στο μεγαλύτερο δυνατό επίπεδο. Μέχρι το 2012 είχαν συνεισφέρει στη δημιουργία του πάνω από 500.000 άνθρωποι. Όλες οι υπηρεσίες του είναι δωρεάν σε αντίθεση με τα Google maps. Η επιλογή έγινε καθαρά με γνώμονα ότι η Google παρέχει περισσότερες δυνατότητες, λειτουργίες, η εμφάνιση είναι πιο ελκυστική και οι χάρτες της έχουν πολλές σχεδιαστικές δυνατότητες.

Για την χρήση της στην RN πρέπει να γίνει μέσω μιας βιβλιοθήκης. Στην συγκεκριμένη εφαρμογή χρησιμοποιήθηκε η react-native-maps (υπάρχει η δυνατότητα μέσω της βιβλιοθήκης να γίνει επιλογή άλλων χαρτών). Πρόκειται για ένα component που αποστέλλει platform-native κώδικα που χρειάζεται να γίνει μετατροπή μαζί με την RN. Τα βασικότερα tags είναι το Map view και το marker, όπου το καθένα δίνουν αντίστοιχα τον χάρτη και το άλλο τα σημεία που επιλέγει ο χρήστης. Ένα σημαντικό αρνητικό έγκειται στο ότι η εγκατάσταση αποτελεί οδύσσεια καθώς πολλοί χρήστες αντιμετωπίζουν πρόβλημα με τις διάφορες εκδόσεις με αποτέλεσμα να είναι αναγκαίες πολλές παραμετροποιήσεις[5]. Η άλλη επιλογή ήταν το mapbox που αποτελεί βάση απόδοσης/ταχύτητας καλύτερη επιλογή άλλα υστερεί σε ποιότητα εικόνων των χαρτών και η βάση χρηστών που βοηθούν στην βελτίωση της βιβλιοθήκης είναι υποδεκαπλάσια.

1.7 REACT NAVIGATION

Οι εφαρμογές για κινητά τηλέφωνα σπάνια αποτελούνται από μία οθόνη. Η διαχείριση της παρουσίασης και της μετάβασης μεταξύ πολλαπλών οθονών γίνεται συνήθως από αυτό που είναι γνωστό ως πλοηγός. Το react navigation παρέχει μια απλή λύση πλοήγησης με την δυνατότητα να παρουσιάζει τόσο *stack* όσο και *tabbed* navigation και για Android και για ios. Είναι μια αυτόνομη βιβλιοθήκη που επιτρέπει στους προγραμματιστές να ρυθμίζουν τις οθόνες της εφαρμογής με λίγες γραμμές κώδικα. Το documentation που υπάρχει είναι τεράστιο με λύσεις για σχεδόν όλα τα προβλήματα. Για την εγκατάσταση του είναι υποχρεωτικό να είναι προ εγκατεστημένες οι βιβλιοθήκες react-native-reanimated, react-native-gesture-handler, react-native-screens, react-native-safe-area-context, @react-native-community/masked-view[4]. Το react-navigation παρέχει πολλούς τρόπους navigation. Στο ThessParking χρησιμοποιήθηκαν ο *stack* navigator και ο *drawer* navigator.

Ο *stack* navigator παρέχει έναν τρόπο για να μεταβαίνει η εφαρμογή μεταξύ οθονών όπου κάθε νέα οθόνη τοποθετείται πάνω από μία στοίβα. Από προεπιλογή, *stack* navigator έχει ρυθμιστεί ώστε να έχει τη γνωστή εμφάνιση και αίσθηση όπως σε ios και Android: οι νέες οθόνες σύρονται από τα δεξιά στο ios, ξεθωριάζουν από κάτω στο Android. Για να χρησιμοποιήσει κάποιος τον *stack* navigator, δημιουργεί την συνάρτηση createNativeStackNavigator() που επιστρέφει ένα αντικείμενο που έχει 2 ιδιότητες Screen και Navigator που χρησιμοποιούνται για να γίνεται εύκολη πλοήγηση του χρήστη στην εφαρμογή. Επίσης οι Stack.screens πρέπει να περιέχονται σε ένα navigation Container, είναι ένα στοιχείο που διαχειρίζεται το δέντρο πλοήγησης και περιέχει την κατάσταση πλοήγησης. Συνήθως δημιουργείται στο App.js που είναι και η ‘ρίζα’ της εφαρμογής.

Ο Drawer navigator είναι ένας άλλος τρόπος μετάβασης σε άλλη οθόνη, που είναι μάλιστα πολύ δημοφιλής στην ανάπτυξη εφαρμογών Ο χρήστης για να πάει σε άλλη οθόνη πρέπει να σύρει από αριστερά την εφαρμογή για να δει τις επιλογές στις οθόνες που υπάρχουν. Επίσης αυτό μπορεί να συμβεί εάν κάνει κλικ σε κάποιο κουμπί. Για την δημιουργία του, ακολουθείται ακριβώς η ίδια λογική με τον stackNavigator απλά εδώ η συνάρτηση ονομάζεται createDrawerNavigator (ο navigation Container χρησιμοποιείται σε όλες τις περιπτώσεις). Οι δυνατότητες σχεδίασης είναι πολλές καθώς ο χρήστης μπορεί να βάλει μέχρι και εικονίδια δίπλα στον κάθε navigator.

Τέλος η άλλη επιλογή ήταν η βιβλιοθήκη react-native-navigation όπου έχει κάποιες διαφορές. Εδώ να τονιστεί ότι δεν έχει compatibility εάν μεταφερθεί αργότερα η εφαρμογή σε web μορφή αλλά και ούτε με το expo. Το react-native-navigation χρησιμοποιεί native modules και η απόδοσή του είναι συνήθως καλύτερη. Ενώ το react-navigation είναι μια απλή και ισχυρή λύση που παρέχεται από την ίδια την RN. Είναι μία ολοκληρωμένη εφαρμογή σε js σε αντίθεση με την άλλη που χρησιμοποιεί native support που μπορεί να είναι περίπλοκη.

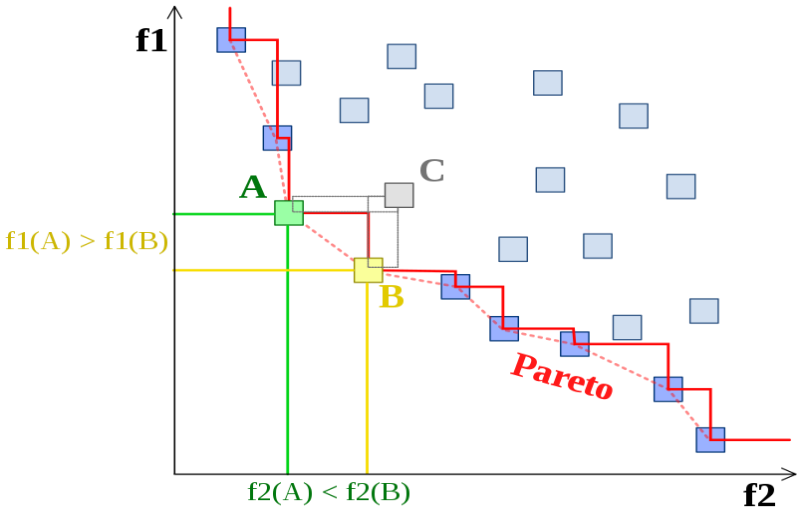
1.8 Skyline Operator

Για την εύρεση των βέλτιστων χώρων στάθμευσης, η εφαρμογή χρησιμοποιεί μία συνάρτηση που βασίζεται στο *Skyline Operator*. Ο *Skyline operator* βοηθάει στη λύση προβλημάτων βελτιστοποίησης και επιλέγει queries για να φιλτράρει τα αποτελέσματα από μία βάση δεδομένων, ώστε να διατηρήσει μόνο εκείνα τα αντικείμενα που δεν είναι χειρότερα από άλλα. Η λειτουργία του βασίζεται στο pareto front που αναλύεται στη συνέχεια.

1.8.1 Pareto front

Pareto efficiency ή Pareto optimality είναι μια κατάσταση όπου κανένα κριτήριο δεν μπορεί να είναι καλύτερο από ένα άλλο χωρίς να υπάρχει ένα κριτήριο που είναι χειρότερο από αυτό. Μία κατάσταση

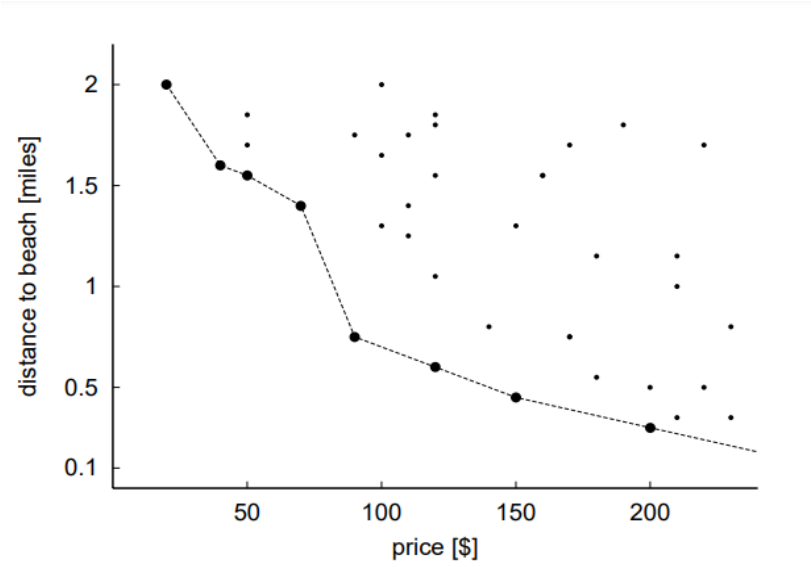
pareto είναι μία βελτιωμένη έκδοση της αρχικής κατάστασης. Μία κατάσταση ονομάζεται Pareto dominated εάν υπάρχει πιθανή βελτίωση του pareto, ενώ εάν καμία αλλαγή δεν μπορεί να οδηγήσει σε βελτιωμένο αποτέλεσμα ονομάζεται Pareto optimal. Το *Pareto front* είναι το σύνολο των αποδοτικών κατανομών pareto που εμφανίζονται. Για ένα σύστημα το *Pareto front* είναι το σύνολο των παραμετροποιήσεων που είναι αποδοτικές, και είναι κάτι πολύ χρήσιμο ιδιαίτερα για τους μηχανικούς. Χρησιμοποιώντας όλες τις βέλτιστες λύσεις, μπορεί να κάνει συγκεντρωτικές αντισταθμίσεις στο συγκεκριμένο σύνολο, χωρίς να λαμβάνει υπόψη όλα τα δεδομένα. Ακολουθεί σχήμα που δείχνει πως δουλεύει το pareto.



Σχήμα 1.2 Pareto front

1.8.2 Χρήση του Skyline Operator

Ένα κλασικό παράδειγμα της χρήσης του *skyline operator* είναι όταν ένας χρήστης ψάχνει το κατάλληλο κατάλυμα για διακοπές. Ας υποθέσουμε ότι κάποιος θέλει να πάει διακοπές σε ένα νησί και ψάχνει ξενοδοχείο που να είναι φθηνό και κοντά στην παραλία. Δυστυχώς τα δύο αυτά στοιχεία δεν είναι ανάλογα γιατί όσο πιο κοντά βρίσκεται ένα ξενοδοχείο στην παραλία τόσο πιο ακριβό είναι. Εμάς, μας ενδιαφέρουν τα ξενοδοχεία που δεν είναι χειρότερα από οποιοδήποτε άλλο και στις δύο διαστάσεις δηλαδή τιμή και απόσταση. Για παράδειγμα ένα ξενοδοχείο με τιμή 30€ και απόσταση 800 μέτρα κυριαρχεί ενός με τιμή 100€ και απόσταση 1000 μέτρων. Έτσι παρουσιάζονται τα βέλτιστα και ο χρήστης μπορεί να αποφασίσει ποιο προτιμάει.



Σχήμα 1.3 Skyline Operator

Παρόμοια λειτουργία έχει και η συγκεκριμένη εφαρμογή αφού βρίσκει τα βέλτιστα συγκρίνοντας την απόσταση του παρκινγκ από τον χρήστη, αλλά και την τιμή που έχει. Αν δύο έχουν την ίδια περίπου απόσταση από τον χρήστη τότε θα προτιμηθεί αυτό με την μικρότερη τιμή. Μία μικρή διαφορά στη κατηγοριοποίηση των παρκινγκ είναι ότι η απόσταση παίζει μεγαλύτερο ρόλο από ότι η τιμή (π.χ. ένα παρκινγκ βρίσκεται εκατό μέτρα πιο κοντά και να είναι ακριβότερο πενήντα λεπτά, ίσως ο χρήστης να προτιμήσει αυτό καθώς η διαφορά στην τιμή είναι πολύ μικρή), σε αντίθεση με τα ξενοδοχεία[16].

1.9 Επίλογος

Συνοψίζοντας, τα 2 βασικότερα στοιχεία της εφαρμογής η γλώσσα προγραμματισμού και η βάση δεδομένων έγιναν με γνώμονα την χρηστικότητα και την καταλληλόλητα για την συγκεκριμένη εφαρμογή. Το να μπορείς να γράφεις μόνο σε μία γλώσσα προγραμματισμού που είναι ευρέως γνωστή (JavaScript) για 2 πλατφόρμες κινητών αποτελεί τεράστιο πλεονέκτημα κυρίως χρονικό. Επιπλέον το σωστό documentation στις περισσότερες βιβλιοθήκες μαζί με την μεγάλη κοινότητα που υπάρχει για να προσφέρει βοήθεια, κάνουν την RN ιδεώδη επιλογή. Τέλος για το firebase η επιλογή του αφορά στο ότι πρόκειται για μία γρήγορη, ασφαλή και πολυπρόσωπη επιλογή με χιλιάδες δυνατότητες από authentication του χρήστη μέχρι διαφήμιση των πελατών.

Κεφάλαιο 2ο: Πολιτική Απορρήτου

2.1 Εισαγωγή

Στις μέρες μας οι περισσότερες εφαρμογές χρησιμοποιούν διάφορα δεδομένα που παίρνουν από τους χρήστες για διαφόρους λόγους π χ. Βελτίωση της εμπειρίας του χρήστη. Για την αποφυγή της χρήσης αυτών των δεδομένων για αισχροκέρδεια και γιατί κάποια από τα δεδομένα αυτά περιέχουν πολύ προσωπικές πληροφορίες, οι περισσότερες χώρες έχουν θεσπίσει νομοθετικά πλαίσια πάνω στα οποία, όλες οι εφαρμογές πρέπει να πληρούν κάποιες προϋποθέσεις όσον αφορά τα προσωπικά δεδομένα που διαχειρίζονται, ώστε να αποτραπεί η χρήση αυτών για άλλους λόγους. Ακόμα και αν η εφαρμογή δεν συλλέγει άμεσα προσωπικά δεδομένα, ενδέχεται να εξακολουθεί να χρειάζεται πολιτική απορρήτου εάν χρησιμοποιεί εργαλεία όπως το Google analytics που χρησιμοποιούν πολλές εφαρμογές. Τα προσωπικά δεδομένα μπορούν να λάβουν πολλές μορφές. Μπορεί να είναι το όνομα του χρήστη, το email, ο αριθμός τηλεφώνου ή φυσική διεύθυνση. Μπορεί επίσης να είναι λιγότερο προφανείς τύποι δεδομένων όπως η διεύθυνση IP, δεδομένα καταγραφής και πληροφορίες που συλλέγονται μέσω cookies.

2.2 Νόμοι για το πολιτικό απόρρητο σε όλο τον κόσμο

Οι Ηνωμένες Πολιτείες είναι μία από της λίγες χώρες χωρίς πολιτική σε εθνικό ή ομοσπονδιακό επίπεδο που επιβάλλει η Πολιτική Απορρήτου. Ωστόσο, ο νόμος για την προστασία της ιδιωτικής ζωής της Καλιφόρνιας στο Διαδίκτυο (CalOPPA) αναφέρει ότι εάν η εφαρμογή ή ο ιστότοπος συλλέγει προσωπικά αναγνωρίσιμα δεδομένα από κατοίκους της πολιτείας της Καλιφόρνιας, πρέπει να υπάρχει Πολιτική Απορρήτου. Δεδομένου ότι είναι πολύ πιθανό η εφαρμογή που απευθύνεται σε κοινό που διαμένει στις Η.Π.Α., να χρησιμοποιηθεί από έναν κάτοικο της Καλιφόρνιας, ανεξάρτητα από το που στον κόσμο που βρίσκεται, το CalOPPA καταλήγει να έχει πολύ μεγάλη εμβέλεια. Στον Καναδά, ο νόμος περί προστασίας προσωπικών πληροφοριών και ηλεκτρονικών εγγράφων (PIPEDA) καθορίζει τις απαιτήσεις για τον τρόπο με τον οποίο οι οργανισμοί πρέπει να χειρίζονται τις προσωπικές πληροφορίες των Καναδών κατοίκων. Ο Γενικός Κανονισμός Προστασίας Δεδομένων της Ευρωπαϊκής Ένωσης (GDPR) που εισήχθη το 2018 είναι ένας από τους ισχυρότερους νόμους για την προστασία των προσωπικών πληροφοριών των ατόμων και έχει παγκόσμια εμβέλεια. Η Ιαπωνία έχει θεσπίσει ισχυρή προστασία για τους κατοίκους της. Ο νόμος περί προστασίας των προσωπικών πληροφοριών που τέθηκε σε ισχύ τον Μάιο του 2017 περιέχει παρόμοιες διατάξεις με αυτές που περιγράφονται στον GDPR της ευρωπαϊκής ένωσης. Ο νόμος περί απορρήτου της Αυστραλίας του 1988 προστατεύει τα προσωπικά δεδομένα των Αυστραλών πολιτών. Ακόμη και αν δεν δοθεί τόσο σημασία στις νομικές ευθύνες που αναφέρονται παραπάνω, είναι καλό για τη διαφάνεια και τη φήμη του οργανισμού-εφαρμογής να έχει μια Πολιτική Απορρήτου που καθορίζει με σαφήνεια τον τρόπο με τον οποίο χρησιμοποιούνται τα προσωπικά δεδομένα των χρηστών.

2.3 Τι πρέπει να περιλαμβάνεται σε μία πολιτική απορρήτου

Η πολιτική απορρήτου θα πρέπει να ενημερώσει τους χρήστες για:

- Τις μορφές των προσωπικών πληροφοριών που συλλέγονται
- Πως συλλέγονται τα δεδομένα αυτά
- Πως οι χρήστες μπορούν να ζητήσουν περισσότερες λεπτομέρειες σχετικά με τις πληροφορίες που συλλέγονται

- Γιατί σκοπεύει η εφαρμογή να χρησιμοποιήσει τις προσωπικές πληροφορίες
- Τυχόν τρίτα μέρη τα οποία συλλέγουν προσωπικές πληροφορίες μέσω της εφαρμογής

Πολλές εφαρμογές τρίτων επιμένουν να αποκαλύψει ο χρήστης ότι τις χρησιμοποιεί και αυτό συνεπάγεται ότι θα συλλέγουν τα προσωπικά τους δεδομένα. Για παράδειγμα το Google analytics που χρησιμοποιείται από το ThessParking απαιτεί τους χρήστες του να το κάνουν αυτό στους Όρους Παροχής Υπηρεσιών (ακολουθεί απόσπασμα από τους Όρους Παροχής Υπηρεσιών του Google Analytics):

Απαγορεύεται να διαβιβάζετε, καθώς και να βοηθάτε ή να επιτρέπετε σε οποιοδήποτε τρίτο μέρος να διαβιβάζει στην Google πληροφορίες, οι οποίες θα μπορούσαν να χρησιμοποιηθούν ή να αναγνωριστούν από την Google ως στοιχεία προσωπικής ταυτοποίησης. Οφείλετε να διαθέτετε κατάλληλη Πολιτική απορρήτου, την οποία θα τηρείτε, καθώς και να συμμορφώνεστε με όλους τους ισχύοντες νόμους, πολιτικές και κανονισμούς σχετικά με τη συλλογή πληροφοριών από Χρήστες. Οφείλετε να αναρτήσετε μια Πολιτική απορρήτου, η οποία θα πρέπει να ενημερώνει για την από μέρους σας χρήση cookie, αναγνωριστικών για κινητές συσκευές (π.χ. Αναγνωριστικό διαφήμισης Android ή Αναγνωριστικό διαφήμισης για iOS) ή παρόμοιας τεχνολογίας για τη συλλογή δεδομένων. Οφείλετε να γνωστοποιήσετε τη χρήση του Google Analytics, καθώς και τον τρόπο με τον οποίο η Υπηρεσία συλλέγει και επεξεργάζεται δεδομένα. Αυτό μπορεί να επιτευχθεί προβάλλοντας έναν εμφανή σύνδεσμο προς τη σελίδα "Πώς χρησιμοποιεί η Google πληροφορίες από ιστοτόπους ή εφαρμογές που χρησιμοποιούν τις υπηρεσίες μας", (η οποία βρίσκεται στη διεύθυνση www.google.com/policies/privacy/partners/, ή σε οποιαδήποτε άλλη διεύθυνση URL που ενδεχομένως παρέχει η Google κατά καιρούς). Οφείλετε να καταβάλλετε εμπορικά εύλογες προσπάθειες, προκειμένου να εξασφαλίζετε ότι ο Χρήστης ενημερώνεται με σαφείς και αναλυτικές πληροφορίες, καθώς και ότι συναινεί στην αποθήκευση και στην πρόσβαση σε cookie ή άλλες πληροφορίες στη συσκευή του Χρήστη, σε περιπτώσεις όπου αυτή η δραστηριότητα συνδέεται με την Υπηρεσία και οι εν λόγω πληροφορίες και η λήψη συναίνεσης απαιτούνται από τον νόμο.

Απαγορεύεται να παρακάμπτετε οποιεσδήποτε λειτουργίες προστασίας απορρήτου (π.χ. εξαίρεση) που αποτελούν μέρος της Υπηρεσίας. Οφείλετε να συμμορφώνεστε με όλες τις ισχύουσες πολιτικές του Google Analytics που βρίσκονται στη διεύθυνση www.google.com/analytics/policies/ (ή σε άλλο URL που ενδέχεται να παρέχει η Google), όπως τροποποιούνται κατά καιρούς (οι "Πολιτικές του Google Analytics").

Μπορείτε να συμμετέχετε σε μια ολοκληρωμένη έκδοση του Google Analytics και σε συγκεκριμένες υπηρεσίες διαφήμισης της Google ("Λειτουργίες διαφήμισης του Google Analytics"). Αν χρησιμοποιείτε τις Λειτουργίες διαφήμισης του Google Analytics, οφείλετε να συμμορφώνεστε με την πολιτική για τις λειτουργίες διαφήμισης του Google Analytics (διαθέσιμη στη διεύθυνση support.google.com/analytics/bin/answer.py?hl=en&topic=2611283&answer=2700409). Η από μέρους σας πρόσβαση σε οποιαδήποτε υπηρεσία διαφήμισης της Google και η χρήση αυτής υπόκειται στους ισχύοντες όρους που έχουν συναφθεί ανάμεσα σε Εσάς και την Google και αφορούν στη συγκεκριμένη υπηρεσία.

Αν χρησιμοποιείτε την Αρχική σελίδα Platform, η από μέρους σας χρήση αυτής υπόκειται στους Πρόσθετους όρους της Αρχικής σελίδας (ή όπως θα μετονομαστούν στη συνέχεια), οι οποίοι είναι διαθέσιμοι στη διεύθυνση https://support.google.com/marketingplatform/answer/9047313 (ή σε άλλο URL που τυχόν θα παρέχει η Google), όπως τροποποιούνται κατά καιρούς (οι "Όροι της Αρχικής σελίδας Platform").

Στον βαθμό κατά τον οποίο η χρήση της Υπηρεσίας από Εσάς εμπίπτει στο πεδίο εφαρμογής, Εσείς και η Google συμφωνείτε με τους Όρους επεξεργασίας δεδομένων διαφημίσεων Google που αναφέρονται στη διεύθυνση https://privacy.google.com/businesses/processorterms (οι "Όροι επεξεργασίας"). Η Google δεν θα τροποποιήσει τους Όρους επεξεργασίας, εκτός αν αυτό επιτρέπεται ρητά σύμφωνα με τους Όρους επεξεργασίας[17].

2.4 Απαιτήσεις App Store

Εκτός από τους νόμους οι εφαρμογές πρέπει να ακολουθούν τους κανόνες-απαιτήσεις των καταστημάτων εφαρμογών. Το Apple App Store, το Google play και άλλα, έχουν Όρους και Προϋποθέσεις που θα πρέπει να ‘συμμορφωθούν’ οι εφαρμογές προκειμένου να καταχωρηθούν στα καταστήματα. Παρακάτω θα αναλυθούν οι απαιτήσεις της Πολιτικής Απορρήτου για κάθε ένα από τα καταστήματα με πιο εκτενή αναφορά στο Google play.

Από τις 3 Οκτωβρίου 2018, η Apple επιβεβαίωσε ότι όλες οι νέες εφαρμογές ή ενημερωμένες εφαρμογές θα απαιτούν Πολιτική Απορρήτου, ανεξάρτητα από το αν η εφαρμογή συλλέγει ή όχι προσωπικές πληροφορίες. Εάν η εφαρμογή είναι καταχωρημένη στο App Store τότε πρέπει να ακολουθεί τις οδηγίες που υπάρχουν. Η πολιτική αυτή του Απορρήτου περιλαμβάνει (ακολουθεί απόσπασμα από τα docs του App store):

- Πολιτικές απορρήτου: Όλες οι εφαρμογές πρέπει να περιλαμβάνουν έναν σύνδεσμο προς την πολιτική απορρήτου τους στο πεδίο μεταδεδομένων του App Store Connect και εντός της εφαρμογής με εύκολο προσβάσιμο τρόπο. Η πολιτική απορρήτου πρέπει σαφώς και ρητά: Να προσδιορίσει ποια δεδομένα, εάν υπάρχουν συλλέγει η εφαρμογή, πως τα συλλέγει αυτά και όλες τις χρήσεις των δεδομένων αυτών. Οποιαδήποτε τρίτη εταιρία με την οποία μια εφαρμογή μοιράζεται δεδομένα χρήστη όπως εργαλεία ανάλυσης, διαφημιστικά δίκτυα και SDK τρίτων, καθώς και οποιαδήποτε μητρική, θυγατρική ή άλλες σχετικές οντότητες που θα έχουν πρόσβαση σε χρήστες δεδομένα - θα παρέχουν την ίδια ή ίση προστασία των δεδομένων χρήστη, όπως αναφέρεται στην πολιτική απορρήτου της εφαρμογής και απαιτείται από αυτές τις Οδηγίες. Πρέπει να εξηγεί η εφαρμογή τις πολιτικές διατήρησης / διαγραφής δεδομένων και να περιγράφει πώς ένας χρήστης μπορεί να ανακαλέσει τη συγκατάθεσή του και / ή να ζητήσει τη διαγραφή δεδομένων χρήστη.

- Άδειες: Οι εφαρμογές που συλλέγουν δεδομένα χρήστη ή χρήσης πρέπει να εξασφαλίζουν τη συγκατάθεση του για τη συλλογή δεδομένων, ακόμη και αν αυτά τα δεδομένα θεωρούνται ανώνυμα κατά τη στιγμή ή αμέσως μετά τη συλλογή. Η πληρωμένη λειτουργικότητα δεν πρέπει να εξαρτάται ή να απαιτεί από έναν χρήστη να παραχωρήσει πρόσβαση σε αυτά τα δεδομένα. Οι εφαρμογές πρέπει επίσης να παρέχουν στον πελάτη έναν εύκολα προσβάσιμο και κατανοητό τρόπο ανάκλησης της συγκατάθεσης. Οι εφαρμογές που συλλέγουν δεδομένα για έννομο

συμφέρον χωρίς συγκατάθεση βασιζόμενοι στους όρους του Γενικού Κανονισμού Προστασίας Δεδομένων της Ευρωπαϊκής Ένωσης («GDPR») ή παρόμοιο καταστατικό πρέπει να συμμορφώνονται με όλους τους όρους του εν λόγω νόμου.

• Ελαχιστοποίηση δεδομένων: Οι εφαρμογές πρέπει να ζητούν πρόσβαση μόνο σε δεδομένα που σχετίζονται με την κύρια λειτουργικότητα της εφαρμογής και πρέπει να συλλέγουν και να χρησιμοποιούν μόνο δεδομένα που απαιτούνται για την εκτέλεση της σχετικής εργασίας. Όπου είναι δυνατόν, χρησιμοποιήστε ένα στοιχείο κοινής χρήσης αντί να ζητήσετε πλήρη πρόσβαση σε προστατευμένους πόρους, όπως Φωτογραφίες ή Επαφές[6].

2.5 Πολιτικές απορρήτου για εφαρμογές Android

Η συμφωνία διανομής προγραμματιστή Google Play αναφέρει ότι εάν υπάρχει διάθεση της εφαρμογή μέσω του καταστήματος Google Play, πρέπει:

- Να προστατευτεί το απόρρητο και τα νόμιμα δικαιώματα των χρηστών.
- Να υπάρχει ενημέρωση στους χρήστες ότι τα προσωπικά δεδομένα θα χρησιμοποιηθούν από την εφαρμογή
- Να υπάρχει νομικά επαρκή σημείωση απορρήτου και προστασία για αυτούς τους χρήστες

Μια "νομικά επαρκής σημείωση απορρήτου" είναι μια Πολιτική απορρήτου, οπότε εάν η εφαρμογή συλλέγει προσωπικές πληροφορίες σχετικά με τους χρήστες της, χρειάζεστε μία σύμφωνα με την Google. Ακολουθεί απόσπασμα από το συμφωνητικό διανομής του Google Play:

Συμφωνείτε ότι εάν διαθέτετε τα Προϊόντα σας μέσω του Google Play, θα προστατεύσετε το απόρρητο και τα νομικά δικαιώματα των χρηστών. Εάν οι χρήστες σάς παρέχουν ή εάν το Προϊόν σας χρησιμοποιεί ή έχει πρόσβαση σε ονόματα χρηστών, κωδικούς πρόσβασης ή άλλα προσωπικά στοιχεία ή στοιχεία σύνδεσης, συμφωνείτε ότι θα ενημερώνετε τους χρήστες ότι οι πληροφορίες θα είναι διαθέσιμες στο Προϊόν σας. Επίσης, συμφωνείτε ότι θα παρέχετε τη δέουσα, από νομική άποψη, σημείωση απορρήτου και προστασία για τους χρήστες αυτούς. Επιπλέον, το Προϊόν σας μπορεί να χρησιμοποιεί αυτές τις πληροφορίες μόνο για τους συγκεκριμένους σκοπούς για τους οποίους ο χρήστης σάς έχει παραχωρήσει την άδειά του. Εάν το Προϊόν σας αποθηκεύει προσωπικές ή ευαίσθητες πληροφορίες που παρέχονται από χρήστες, συμφωνείτε ότι αυτό θα γίνεται με ασφάλεια και μόνο για το χρονικό διάστημα που απαιτείται. Ωστόσο, εάν ο χρήστης έχει επιλέξει να συνάψει ξεχωριστό συμφωνητικό μαζί σας, το οποίο επιτρέπει σε εσάς ή στο Προϊόν σας την αποθήκευση ή τη χρήση προσωπικών ή ευαίσθητων πληροφοριών που σχετίζονται άμεσα με το Προϊόν σας (χωρίς να συμπεριλαμβάνονται άλλα προϊόντα ή εφαρμογές), τότε η χρήση των εν λόγω πληροφοριών θα διέπεται από τους όρους αυτού του ξεχωριστού συμφωνητικού. Εάν ο χρήστης παρέχει στο προϊόν σας στοιχεία του Λογαριασμού Google, το Προϊόν σας μπορεί να χρησιμοποιήσει τα στοιχεία αυτά μόνο για πρόσβαση στον Λογαριασμό Google του χρήστη όταν, και αποκλειστικά για τους σκοπούς για τους οποίους, ο χρήστης σάς έχει δώσει την άδεια να το κάνετε.

Το Κέντρο Πολιτικής Προγραμματιστών της Google περιλαμβάνει μια ενότητα Απορρήτου, Ασφάλειας και Απάτης που απαιτεί επίσης μια Πολιτική Απορρήτου (ακολουθεί κείμενο από την ιστοσελίδα της Google):

Προσωπικές και ευαίσθητες πληροφορίες

Τα προσωπικά και ευαίσθητα δεδομένα χρήστη περιλαμβάνουν, ενδεικτικά, στοιχεία προσωπικής ταυτοποίησης, οικονομικά στοιχεία και στοιχεία πληρωμής, πληροφορίες ελέγχου ταυτότητας, δεδομένα που σχετίζονται με τον τηλεφωνικό κατάλογο, τις επαφές, την τοποθεσία συσκευής, τα SMS και τις κλήσεις, το απόθεμα άλλων εφαρμογών στη συσκευή, τα δεδομένα μικροφώνου, κάμερας και άλλα ευαίσθητα δεδομένα συσκευής ή χρήσης. Εάν η εφαρμογή σας χειρίζεται προσωπικά και ευαίσθητα δεδομένα χρήστη, πρέπει να κάνετε τα εξής:

Να περιορίσετε την από μέρους σας πρόσβαση, συλλογή, χρήση και κοινοποίηση προσωπικών και ευαίσθητων δεδομένων χρήστη που αποκτούνται μέσω της εφαρμογής αποκλειστικά για σκοπούς που σχετίζονται άμεσα με την παροχή και τη βελτίωση των λειτουργιών της εφαρμογής (π.χ. αναμενόμενη λειτουργικότητα χρήστη που τεκμηριώνεται και προωθείται στην περιγραφή της εφαρμογής στο Google Play). Η κοινοποίηση προσωπικών και ευαίσθητων δεδομένων χρήστη περιλαμβάνει τη χρήση SDK ή άλλων υπηρεσιών τρίτων μερών που προκαλούν τη μεταφορά των δεδομένων σε ένα τρίτο μέρος. Οι εφαρμογές που επεκτείνουν τη χρήση προσωπικών και ευαίσθητων δεδομένων χρήστη για την προβολή διαφημίσεων πρέπει να συμμορφώνονται με την Πολιτική διαφήμισης.

Να χειρίζεστε με ασφάλεια όλα τα προσωπικά και ευαίσθητα δεδομένα χρήστη, συμπεριλαμβανομένης της χρήσης σύγχρονων τεχνικών κρυπτογράφησης για τη μετάδοσή τους (π.χ. μέσω HTTPS).

Να χρησιμοποιείτε ένα αίτημα αδειών χρόνου εκτέλεσης (runtime), όποτε αυτό είναι διαθέσιμο, πριν από την πρόσβαση σε δεδομένα που περιορίζονται από άδειες Android.

Να μην πουλάτε τα προσωπικά και ευαίσθητα δεδομένα χρήστη.

Απαίτηση εμφανούς αποκάλυψης και συναίνεσης

Σε περιπτώσεις όπου οι χρήστες μπορεί να μην περιμένουν εύλογα ότι θα απαιτούνται τα προσωπικά και ευαίσθητα δεδομένα χρήστη για την παροχή ή τη βελτίωση των δυνατοτήτων ή των λειτουργιών που συμμορφώνονται με την πολιτική εντός της εφαρμογής σας (π.χ. η συλλογή δεδομένων πραγματοποιείται στο παρασκήνιο της εφαρμογής σας), θα πρέπει να πληροίτε τις ακόλουθες απαιτήσεις:

Πρέπει να υπάρχει μια αναφορά εντός της εφαρμογής σχετικά με τη συλλογή, τη χρήση, την κοινοποίηση δεδομένων και την πρόσβαση σε αυτά. Η αποκάλυψη εντός εφαρμογής:

- Θα πρέπει να βρίσκεται εντός της ίδιας της εφαρμογής και όχι μόνο στην περιγραφή της εφαρμογής ή σε έναν ιστότοπο,
- Θα πρέπει να εμφανίζεται κατά την κανονική χρήση της εφαρμογής και να μην απαιτεί από τον χρήστη να περιηγηθεί σε κάποιο μενού ή ρυθμίσεις,
- Θα πρέπει να περιγράφει τα δεδομένα στα οποία αποκτάται πρόσβαση ή συλλέγονται,
- Θα πρέπει να περιγράφει τον τρόπο χρήσης ή/και κοινής χρήσης των δεδομένων,
- Δεν θα πρέπει να εμφανίζεται μόνο μέσα στην πολιτική απορρήτου ή τους όρους παροχής υπηρεσιών και
- Δεν θα πρέπει να συμπεριληφθεί με τις άλλες αποκαλύψεις που δεν σχετίζονται με τη συλλογή προσωπικών ή ευαίσθητων δεδομένων.

Η αναφορά εντός εφαρμογής θα πρέπει να συνοδεύει και να προηγείται αμέσως ενός αιτήματος συναίνεσης χρήστη και, όπου αυτό είναι διαθέσιμο, μιας σχετικής άδειας χρόνου εκτέλεσης (runtime). Δεν μπορεί να αποκτά πρόσβαση ο προγραμματιστής ή να συλλέγει τυχόν προσωπικά ή ευαίσθητα δεδομένα προτού λάβει τη συναίνεση του χρήστη. Το αίτημα συναίνεσης της εφαρμογής (ακολουθεί κείμενο από την ιστοσελίδα της Google):

- Θα πρέπει να εμφανίζει το παράθυρο διαλόγου συναίνεσης με ξεκάθαρο και σαφή τρόπο.
- Θα πρέπει να απαιτεί καταφατική ενέργεια από τον χρήστη (π.χ. πάτημα για αποδοχή, τικ σε ένα πλαίσιο επιλογής).
- Δεν θα πρέπει να ερμηνεύει την απομάκρυνση από την αποκάλυψη (μεταξύ άλλων το πάτημα σε ένα άλλο σημείο ή το πάτημα του κουμπιού Πίσω ή Αρχική σελίδα) ως συναίνεση και
- Δεν θα πρέπει να χρησιμοποιεί μηνύματα αυτόματης παράλειψης ή λήξης ως μέσο λήψης συναίνεσης από τους χρήστες[7].

2.6 Πολιτική απορρήτου ThessParking

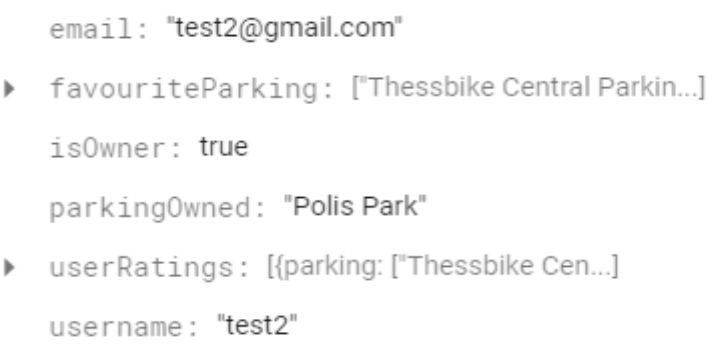
Η εφαρμογή ThessParking αποτελεί από την φύση της, εφαρμογή που χρειάζεται πρόσβαση στα δεδομένα του χρήστη ώστε να είναι λειτουργική. Όπως κάθε εφαρμογή που χρησιμοποιεί χάρτες για να εξυπηρετήσει τον χρήστη να φτάσει στον προορισμό του, πρέπει να έχει πρόσβαση στην τοποθεσία του ώστε να του δείξει την σωστή διαδρομή. Επίσης η εφαρμογή χρησιμοποιεί authentication, έτσι στοιχεία όπως email και κωδικός πρόσβασης είναι απαιτούμενα πεδία ώστε να δημιουργηθεί χρήστης στην εφαρμογή και να έχει πρόσβαση σε όλες τις δυνατότητες της. Παρακάτω αναλύονται τα δεδομένα που συλλέγονται και το πως συλλέγονται.

Αρχικά για το *geolocation* του χρήστη απαιτείται ο χρήστης να έχει συνέχεια ενεργοποιημένη την τοποθεσία του. Εάν ο χρήστης βρίσκεται στους χάρτες και δεν έχει ενεργοποιημένη της τοποθεσία του, εμφανίζεται μήνυμα που προτρέπει να την ενεργοποιήσει ώστε να είναι σε θέση να χρησιμοποιήσει την εφαρμογή. Η εφαρμογή δεν αποθηκεύει κανένα στοιχείο από τις συντεταγμένες του χρήστη, απλά μέσω μιας βιβλιοθήκης εμφανίζεται το σημείο στο οποίο βρίσκεται. Επιπλέον εάν ο χρήστης επιλέξει να ακολουθήσει μια διαδρομή για ένα παρκινγκ, δεν αποθηκεύεται η διαδρομή. Τα μόνα στοιχεία από συντεταγμένες που είναι αποθηκευμένα είναι οι συντεταγμένες των παρκινγκ ώστε να βρίσκονται σωστά τοποθετημένα μέσα στο χάρτη.

Ένα ακόμα στοιχείο που πρέπει να ‘δώσει’ ο χρήστης στην εφαρμογή ώστε να μπορέσει να απολαύσει όλες τις δυνατότητες της, είναι το email του. Εάν ο χρήστης θέλει να αποθηκεύει τα αγαπημένα του παρκινγκ ή να βαθμολογεί οποιοδήποτε παρκινγκ πρέπει να δημιουργήσει λογαριασμό. Κατά την διάρκεια δημιουργίας λογαριασμού, θα του ζητηθεί να επιλέξει ένα όνομα χρήστη, κωδικό πρόσβασης, έγκυρο email και εάν είναι ιδιοκτήτης παρκινγκ ή όχι. Τα στοιχεία email και κωδικός πρόσβασης αποθηκεύονται στη βάση και χρησιμοποιούνται για να κάνει log in ο χρήστης. Ο διαχειριστής δεν μπορεί να δει τους κωδικούς πρόσβασης των χρηστών παρά μόνο τα email τα οποία δεν μεταβιβάζονται σε τρίτες εταιρείες ούτε χρησιμοποιούνται για την προώθηση προϊόντων ή της εφαρμογής.

Εκτός από το authentication τα στοιχεία email, username, αποθηκεύονται σε μία συλλογή στο firestore ονόματι users. Ο λόγος που γίνεται αυτό είναι για την καλύτερη λειτουργία της εφαρμογής και για δίνονται περισσότερες δυνατότητες στους χρήστες. Το πεδίο isOwner χρησιμοποιείται για να

δείξει εάν ο χρήστης είναι ιδιοκτήτης, εάν απαντήσει θετικά μετά από ένα μικρό χρονικό διάστημα θα λάβει ένα email ώστε να δώσει τα διαπιστευτήρια που τον κάνουν ιδιοκτήτη του συγκεκριμένου παρκινγκ. Μετά την διασταύρωση δημιουργείται ένα ακόμη πεδίο parkingOwned όπου σαν τιμή μπαίνει το όνομα το παρκινγκ, και έτσι όταν κάνει log in ο ιδιοκτήτης αυτός θα του εμφανίζονται τα στατιστικά στοιχεία για το συγκεκριμένο παρκινγκ (ακολουθεί εικόνα από το firestore)



Εικόνα 2.1 Πεδία που περιέχει η βάση δεδομένων των χρηστών

Στην παραπάνω εικόνα βλέπουμε δύο πεδία τα οποία δημιουργούνται αφού ο χρήστης κάνει κάποιες ενέργειες π.χ. για να μπει το Thessbike Central Parking στα αγαπημένα του έπρεπε να επιλέξει πάνω στο αστεράκι που υπάρχει στη συγκεκριμένη καρτέλα του παρκινγκ. Κάθε φορά που ο χρήστης επιλέγει να προσθέσει ένα παρκινγκ στα αγαπημένα του αυτομάτως προστίθεται στο πεδίο favouriteParking, δημιουργώντας ένα πίνακα. Οι προτιμήσεις αυτές του χρήστη αποθηκεύονται έτσι ώστε κάθε φορά που θα κάνει log in να εμφανίζονται αυτομάτως αυτά που είχε επιλέξει. Επιπλέον τα στοιχεία αυτά χρησιμοποιούνται σε κάποια στατιστικά που δίνονται στους ιδιοκτήτες αλλά παραμένουν ανώνυμα. Πιο συγκεκριμένα ο παραπάνω ιδιοκτήτης θα μπορεί να δει πόσοι χρήστες έχουν αποθηκεύσει το Polis park στα αγαπημένα τους αλλά δεν θα γνωρίζει ποιοι χρήστες είναι αυτοί. Τέλος στις πολιτικές απορρήτου αναφέρεται ξεκάθαρα ότι χρησιμοποιούνται αυτά τα δεδομένα για αυτή την χρήση.

Το άλλο πεδίο που δημιουργείται όταν ο χρήστης αλληλοεπιδρά με την εφαρμογή είναι το user Ratings. Ένας απλός χρήστης της εφαρμογής μπορεί να δει τα ratings από τα παρκινγκ αλλά δεν μπορεί να βαθμολογήσει ο ίδιος, ενώ αν κάποιος που έχει δημιουργήσει λογαριασμό μπορεί να διαμορφώσει το rating σε κάθε παρκινγκ και κάθε φορά που κάνει log in να του εμφανίζεται το συγκεκριμένο. Για τους χρήστες που δεν είναι συνδεδεμένοι οι τιμές που βλέπουν για τα rating είναι μια σταθερή τιμή που υπάρχει στο πεδίο κάθε παρκινγκ στη βάση(δεν βγαίνει δηλαδή από κάποιο μέσο όρο όλων των βαθμολογιών των χρηστών).

Σε κάθε ιδιοκτήτη παρκινγκ υπάρχει μία καρτέλα με γραφήματα όπου φαίνονται πόσοι χρήστες έχουν κάνει κλικ το παρκινγκ τους την συγκεκριμένη ημερομηνία. Για να δημιουργηθούν τα δεδομένα αυτά, κάθε συνδεδεμένος χρήστης που κάνει κλικ σε οποιοδήποτε παρκινγκ για την συγκεκριμένη ημερομηνία αυξάνει τον counter του παρκινγκ(όσες φορές κάνει κλικ ο μετρητής αυξάνεται). Επειδή αυτά τα δεδομένα αφορούν ενέργειες που κάνουν οι χρήστες στην εφαρμογή και αποθηκεύονται χωρίς πρακτικά να φαίνεται όπως π.χ. για το favourite Parking, στην πολιτική απορρήτου αναφέρεται ξεκάθαρα ότι για να δημιουργήσει κάποιος χρήστης λογαριασμό συναινεί στο να αποθηκεύονται κάποιες ενέργειες του για στατιστική χρήση μόνο μέσα στην εφαρμογή, χωρίς αυτά τα δεδομένα να

δίνονται σε τρίτες εταιρείες ή να χρησιμοποιούνται για την προβολή διαφημίσεων συγκεκριμένου τύπου για τον κάθε χρήστη (Η παρακάτω εικόνα είναι από τα δεδομένα που συλλέχθηκαν για ένα παρκινγκ).

Εικόνα 2.2 Πεδία από την βάση δεδομένων frequent

Στην πολιτική απορρήτου γίνεται και αναφορά στα Google analytics που χρησιμοποιεί η εφαρμογή. Το Google Analytics είναι μια από τις κορυφαίες υπηρεσίες της Google, που παρέχεται δωρεάν και η οποία προσφέρει αναλυτικές πληροφορίες σχετικά με την επισκεψιμότητα μιας ιστοσελίδας/εφαρμογής[8]. Ενδεικτικά, μέσω της συγκεκριμένης υπηρεσίας μπορεί ο προγραμματιστής το πώς οι επισκέπτες έφτασαν στην εφαρμογή του, τι χρόνο διέθεσαν σε ποιες σελίδες, καθώς και πολλές ακόμη πληροφορίες που μπορούν να σας βοηθήσουν κατά την βελτιστοποίηση της εφαρμογής. Παρέχει μια πλήρη εικόνα σχετικά με τα δημογραφικά στοιχεία που αφορούν στους επισκέπτες, όπως είναι για παράδειγμα η περιοχή διαμονής τους και η μητρική τους γλώσσα, στατιστικά στοιχεία σχετικά με το ποσοστό επιστροφής στην εφαρμογή και τον χρόνο πλοήγησης σε αυτήν, πληροφορίες, καθώς και τους τρόπους με τους οποίους κατέληξαν σε αυτήν (keywords, referral links κλπ.). Γίνεται αντιληπτό ότι υπάρχει πρόσβαση σε αρκετά δεδομένα του χρήστη αλλά όλα αυτά τα στατιστικά χρησιμοποιούνται για την καλύτερη εμπειρία του π.χ. εάν κάποιος χρήστης είναι από άλλη χώρα θα προτιμήσει να δει μία διαφήμιση στην γλώσσα του παρά στα ελληνικά. Τέλος αποτελεί και ένα μέσο ώστε να υπάρξει άμεση αύξηση της κερδοφορίας της εφαρμογής (ακολουθεί απόσπασμα από τις πολιτικές απορρήτου του ThessParking).

Η εφαρμογή και οι υπηρεσίες μας για κινητά ενδέχεται να χρησιμοποιούν εργαλεία ανάλυσης τρίτων που χρησιμοποιούν *cookies, web beacons* ή άλλες παρόμοιες τεχνολογίες συλλογής πληροφοριών για τη συλλογή τυπικών πληροφοριών δραστηριότητας και χρήσης στο Διαδίκτυο. Οι πληροφορίες που συλλέγονται χρησιμοποιούνται για τη συλλογή στατιστικών αναφορών σχετικά με τη δραστηριότητα των χρηστών, όπως το πόσο συχνά επισκέπτονται την εφαρμογή και τις υπηρεσίες μας για κινητές συσκευές, ποιες σελίδες επισκέπτονται και για πόσο χρονικό διάστημα κ.λπ. την εφαρμογή και τις υπηρεσίες μας για κινητά.

Η πολιτική απορρήτου για την εφαρμογή ThessParking δημιουργήθηκε με την βοήθεια μιας ιστοσελίδας, όπου μέσα από διάφορες ερωτήσεις για την εφαρμογή, φτιάχνει το καταλληλότερο privacy and policy. Ο χρήστης μπορεί να δει την πολιτική απορρήτου πριν δημιουργήσει λογαριασμό και κατ' επέκταση να συμφωνήσει με την πολιτική αυτή. Παρακάτω παρουσιάζονται οι τρόποι που η εφαρμογή συλλέγει δεδομένα όπως είναι γραμμένα στην πολιτική απορρήτου:

Λαμβάνουμε και αποθηκεύουμε οποιαδήποτε πληροφορία μας δίνετε εν γνώσει μας όταν δημιουργείτε λογαριασμό ή συμπληρώνετε τυχόν διαδικτυακές φόρμες στην Εφαρμογή για κινητά. Όταν απαιτείται, αυτές οι πληροφορίες μπορεί να περιλαμβάνουν τα ακόλουθα:

• Στοιχεία λογαριασμού (όπως όνομα χρήστη, μοναδικό αναγνωριστικό χρήστη, κωδικός πρόσβασης κ.λπ.)

• Στοιχεία επικοινωνίας (όπως διεύθυνση ηλεκτρονικού ταχυδρομείου, αριθμός τηλεφώνου κ.λπ.)

• Δεδομένα γεωεντοπισμού της συσκευής σας (όπως γεωγραφικό πλάτος και γεωγραφικό μήκος)

• Δεδομένα όπως αγαπημένα παρκινγκ ή βαθμολογίες παρκινγκ ίσως χρησιμοποιηθούν για στατιστικές έρευνες

• Ορισμένες από τις πληροφορίες που συλλέγουμε είναι απευθείας από εσάς μέσω της Εφαρμογής. Ωστόσο, ενδέχεται επίσης να συλλέξουμε Προσωπικές Πληροφορίες σχετικά με εσάς από άλλες πηγές, όπως δημόσιες βάσεις δεδομένων και κοινούς εταίρους μάρκετινγκ.

2.7 Επίλογος

Συνοψίζοντας, εάν μια εφαρμογή συλλέγει προσωπικές πληροφορίες από τους χρήστες χρειάζεται Πολιτική απορρήτου για να συμμορφώνεται με το νομοθετικό πλαίσιο ανά τον κόσμο. Όχι μόνο αυτό, αλλά τα καταστήματα εφαρμογών, όπως το Google Play και το App Store της Apple, επιμένουν πλέον στους προγραμματιστές να συμπεριλαμβάνουν τις Πολιτικές απορρήτου στις καταχωρίσεις των καταστημάτων εφαρμογών τους καθώς και στις εφαρμογές τους. Η δημιουργία μιας Πολιτικής Απορρήτου είναι επίσης καλή πρακτική για τη διαφάνεια και για να δουν οι πελάτες ότι η εφαρμογή νοιάζεται για τη διατήρηση των προσωπικών δεδομένων τους με ασφάλεια. Όταν ενημερώνεται η Πολιτική απορρήτου με ουσιαστικές αλλαγές, πρέπει να στέλνουν Ειδοποιήσεις Ενημέρωσης για να είναι ακόμη πιο διαφανείς και συμβατές. Η Πολιτική Απορρήτου πρέπει να βρίσκεται:

- Στα καταστήματα εφαρμογών που θα υπάρχει
- Σε ένα μενού Σχετικά με' μέσα στην εφαρμογή
- Επίσης θα πρέπει να βρίσκεται σε όλη την εφαρμογή σε μέρη όπου:
    - Ζητάει άδεια χρήσης προσωπικών δεδομένων για κάτι
    - Σελίδες εγγραφής και σύνδεσης λογαριασμού
    - Σελίδες πληρωμής

Κεφάλαιο 3ο: Διαδικασία δημιουργίας της εφαρμογής

3.1 Εισαγωγή

Αρχικά κατά την δημιουργία μιας εφαρμογής με react ο χρήστης πρέπει να αποφασίσει εάν θα χρησιμοποιήσει το expo ή όχι. Έπειτα θα πρέπει να βρει και να επιλέξει της βασικές βιβλιοθήκες που θα χρησιμοποιήσει. Στην συνέχεια να περάσει στην εγκατάσταση τους μέσω παραμετροποιήσεων που χρειάζονται για κάθε βιβλιοθήκη και τέλος να ξεκινήσει να γράφει τον απαιτούμενο κώδικα που θα τον οδηγήσει στο αποτέλεσμα που επιθυμεί.

3.2 EXPO CLI VS REACT NATIVE CLI

Το expo είναι ένα framework που χρησιμοποιείται για την δημιουργία εφαρμογών RN. Είναι ένα πακέτο με εργαλεία και υπηρεσίες, το οποίο βοηθάει στην εύκολη δημιουργία εφαρμογών RN. Όταν ξεκινάει η δημιουργία του project, ο προγραμματιστής πρέπει να αποφασίσει εξαρχής εάν θα χρησιμοποιήσει Expo ή όχι. Αυτό συμβαίνει επειδή όλη η αρχιτεκτονική της εφαρμογής βασίζεται στην επιλογή αυτή. Ακολουθούν κάποιοι καλοί λόγοι για την δημιουργία εφαρμογής με την χρήση expo:

- Ο γρηγορότερος τρόπος δημιουργίας RN Apps. Εάν ένα έργο χρειάζεται ταχεία ανάπτυξη για την δημιουργία εφαρμογής πολλαπλών πλατφόρμων, το Expo είναι η καλύτερη επιλογή καθώς έχει πολλά πράγματα προεγκατεστημένα.

- Δεν χρειάζεται ο χρήστης να γνωρίζει Native mobile coding. Δηλαδή δεν θα χρειαστεί ο χρήστης να πειράξει κώδικα σε android ή iOs για να δουλέψει μία εφαρμογή.

- Το Xcode και το Android studio δεν είναι χρησιμοποιούνται, όλη την δουλειά την κάνει το expo.

- Με το Expo, δημοσιοποιούνται γρήγορα οι ενημερώσεις της εφαρμογή, με ευκολία. Δεν χρειάζεται να μεταβεί ο χρήστης στο κατάστημα εφαρμογών για να δημοσιεύει κάθε ενημέρωση. Αυτό είναι ένα χρήσιμο χαρακτηριστικό για γρήγορους κύκλους ανάπτυξης και δοκιμών.

- Τέλος, το Expo είναι δωρεάν και επίσης open-source. Η κοινότητα Expo είναι φιλόξενη και αναπτυσσόμενη

Υπάρχουν κάποιες περιπτώσεις που μπορεί το expo να μην είναι το κατάλληλο εργαλείο για την εφαρμογή. Αυτές μπορεί να είναι:

- Το μεγαλύτερο πλεονέκτημα της χρήσης του Expo είναι ότι δεν χρειάζεται να πειραχτεί ο native code. Εάν όμως ο προγραμματιστής ή η ομάδα που εργάζεται γνωρίζει από native code δεν χρειάζεται το Expo.

- Προσθέτει ακόμη ένα στρώμα *abstraction*, που σε πολλούς προγραμματιστές δεν αρέσει αυτό. Η ίδια η RN έχει ένα τέτοιο στρώμα γύρω από στοιχεία του iOS και Android. Η προσθήκη ακόμη ενός επιπέδου δεν είναι κάτι που βολεύει πολλούς προγραμματιστές. Δηλαδή

υπάρχουν πολλά πράγματα που συμβαίνουν στο background χωρίς να υπάρχει έλεγχος από τον προγραμματιστή, κάτι που είναι αρκετά σοβαρό.

- Η επίσημη τεκμηρίωση της Expo αναφέρει ότι δεν είναι ακόμη διαθέσιμα όλα τα API iOS και Android. Λειτουργίες όπως bluetooth, αγορές εντός εφαρμογής δεν είναι διαθέσιμες με το Expo . Έτσι πολλές δυνατότητες ίσως δεν μπορέσει να τις αξιοποιήσει ο προγραμματιστής με το Expo.

- Δεν είναι δυνατή η επιλογή της υπηρεσίας ειδοποιήσεων push. Με το Expo, δεν μπορεί να επιλέξει την υπηρεσία ειδοποιήσεων push, όπως το Firebase. Ο χρήστης μπορεί να χρησιμοποιήσει One Signal. Προφανώς η χρήση firebase είναι καίρια για το project και ένας από τους κύριους λόγους που απορρίφθηκε το expo

Τέλος, ακόμη και εάν γίνεται η χρήση expo μπορεί πάντα ο προγραμματιστής να προχωρήσει στην απεγκατάστασή του, κάνοντας τα εξής βήματα:

- Συνήθως χρειάζεται να ξαναγραφτεί ο κώδικας για push notification
- Χρήση άλλων βιβλιοθηκών καθώς δεν είναι όλες συμβατές με αυτές του expo
- Γενική αναδιαμόρφωση του κώδικα

Η χρήση ή όχι του Expo για τις εφαρμογές RN εξαρτάται από τις δυνατότητες, τα χρονοδιαγράμματα, τις προτιμήσεις και τους προγραμματιστές που εργάζονται σε αυτό[9].

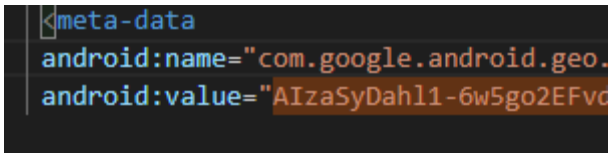
3.3 Έρευνα για παρόμοιες εφαρμογές

Αφού δημιουργήθηκε το project έπρεπε να βρεθούν παραδείγματα άλλων εφαρμογών ώστε να δοθούν ιδέες για την πιο εύκολη υλοποίησή. Η εφαρμογή που είχε παρόμοια φιλοσοφία με το ThessParking είναι :Parkopedia όπου περιέχει εκατομμύρια παρκινγκ από όλο τον κόσμο και πολλές πληροφορίες για το κάθε παρκινγκ. Έτσι έγινε προσπάθεια επικοινωνίας με την συγκεκριμένη εταιρία ώστε εάν ήταν εφικτό να παρέχει πρόσβαση στο API τους. Δηλαδή να μην χρειάζεται η δημιουργία μιας βάσης για κάθε παρκινγκ από την αρχή. Μετά από επικοινωνία τόσο με την συγκεκριμένη εταιρία όσο και με άλλες εταιρείες που παρείχαν API, υπήρχε χρηματικό ποσό που έπρεπε να δοθεί για την πρόσβαση του χρήστη στις υπηρεσίες τους. Έτσι η κατασκευή βάσης με παρκινγκ μόνο για την Θεσσαλονίκη ήταν μονόδρομος.

3.4 Google Map Screen & OpenStreetMaps Screen

3.4.1 Χάρτες της Google

Το GoogleMapScreen αποτελεί την κυριότερη οθόνη της εφαρμογής και αυτή που δημιουργήθηκε πρώτη. Για να έχει πρόσβαση ο χρήστης στους χάρτες της Google, θα πρέπει να δημιουργήσει λογαριασμό στο cloud και να ενεργοποιήσει το API για τους χάρτες. Στο AndroidManifest.xml θα πρέπει να γίνει η εισαγωγή του μοναδικού API key που έχει ο χρήστης.



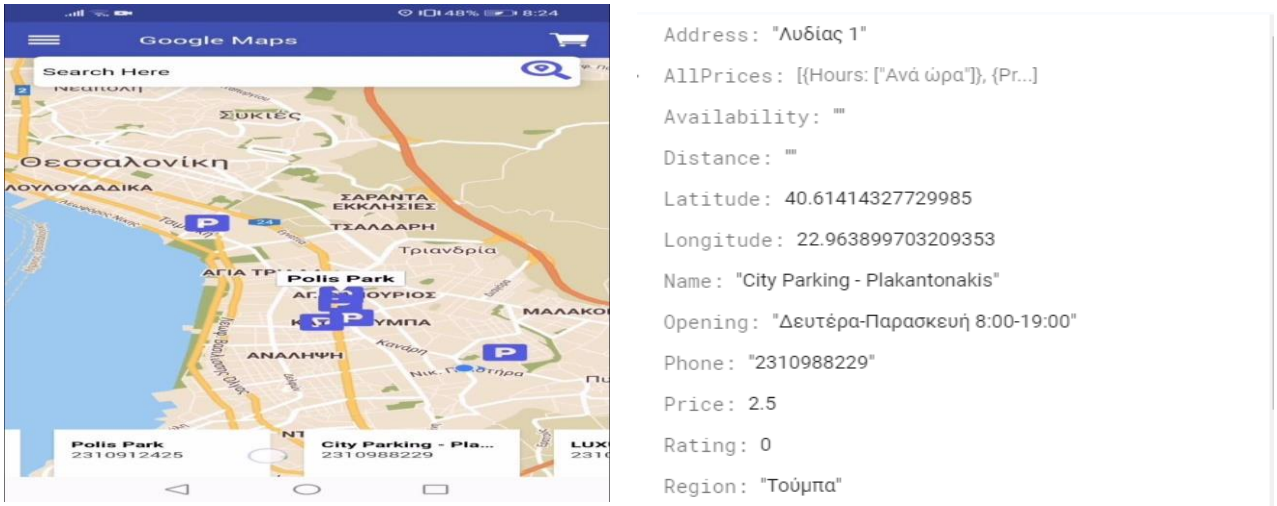
Εικόνα 3.1 Πως γίνεται η πρόσβαση στους χάρτες

Όπως αναφέρθηκε στην αρχή η εγκατάσταση της συγκεκριμένης βιβλιοθήκης είναι λίγο δύσκολη γιατί πρέπει να γίνουν κάποιες αλλαγές στο setting.grandle και στο build.grandle ώστε να γίνει η σωστή εισαγωγή της. Παρόλη την δυσκολία που υπάρχει στην εγκατάσταση στο documentation υπάρχουν λύσεις για τυχόν περιπτώσεις που δεν δουλεύει η βιβλιοθήκη (μαύρη οθόνη,runtime errors κλπ..). Όλες αυτές οι αλλαγές γίνονται εάν η εφαρμογή τρέχει μόνο σε android, για iOS θα πρέπει να γίνουν και άλλες διαφοροποιήσεις native code της εφαρμογής. Αφού ολοκληρώθηκαν οι ρυθμίσεις έγινε η εισαγωγή της βιβλιοθήκης react-native-maps, στην συνέχεια πραγματοποιήθηκε η δημιουργία του χάρτη μαζί με κάποια markers τα οποία τοποθετήθηκαν τυχαία. Στο MapView tag ορίστηκαν αρχικά οι συνταγμένες έτσι ώστε να εμφανίζεται ο χάρτης στην πόλη της Θεσσαλονίκης, ενώ παράλληλα δημιουργήθηκε ένα συγκεκριμένο style για τον χάρτη μέσω από την πλατφόρμα της Google αλλάζοντας ελαφρώς τον καμβά των χρωμάτων, αλλά και πολλά markers που υπήρχαν όπως καφετέριες, μαγαζιά αποκρύφτηκαν από τον χάρτη έτσι ώστε ο χρήστης να εστιάζει μόνο στα παρκινγκ markers.

3.4.2 OpenStreetmap χάρτες

Εκτός από το GoogleMapScreen, δημιουργήθηκε και το OpenstreetMapScreen ώστε να γίνει σύγκριση με το πρώτο και να επιλεχθεί το καταλληλότερο. Η δημιουργία του ήταν δυσκολότερη καθώς δεν αποτελεί την default επιλογή και πρέπει να εισαχθεί στο urlTile κάτι το οποίο χρησιμοποιείται για εισαγωγή άλλων χαρτών. Η παραπάνω λειτουργία δεν δούλεψε, καθώς πολλοί χρήστες android δεν μπορούν να χρησιμοποιήσουν OSM με αυτή την βιβλιοθήκη(με iOs δεν υπάρχει κανένα θέμα λειτουργικότητας), με αποτέλεσμα να γίνει δοκιμή μιας τροποποιημένης βιβλιοθήκης της react-native-maps όπου χρησιμοποιεί σαν default OpenStreetMaps, παρόλα αυτά το αποτέλεσμα δεν ήταν τόσο ελκυστικό, με αποτέλεσμα την χρήση Google maps. Εκτός από αυτό, δεν δίνεται η δυνατότητα τροποποίησης του style του χάρτη, δηλαδή δεν μπορούν να αποκρυφτούν άλλα markers ή να αλλάξει το χρώμα του χάρτη.

Αφού ολοκληρώθηκε η σύγκριση μεταξύ των δύο χαρτών, η προσοχή στράφηκε στο style των markers. Όπου έγινε η εισαγωγή της βιβλιοθήκης react-native-vector-icons όπου υπάρχουν υπεραρκετά εικονίδια που μπορούν να χρησιμοποιηθούν. Έτσι επιλέχθηκε ένα εικονίδιο παρκινγκ προσαρμόστηκε στο χρώμα και το μέγεθος και προστέθηκε στην εφαρμογή. Για να γίνει όμως ‘σωστή’ εισαγωγή των markers έπρεπε να δημιουργηθεί μία βάση δεδομένων όπου θα παίρνει η εφαρμογή τα στοιχεία από εκεί και θα τα εισάγει στο maps. Με την βοήθεια του firebase-firestore δημιουργήθηκε η βάση με κάποια πεδία που δίνουν διάφορα στοιχεία και έγινε η εισαγωγή των παρκινγκ. Τα περισσότερα στοιχεία για τα παρκινγκ αντλούνται από το parkopedia και το google maps (εικόνα από το database και αρχικό στάδιο εφαρμογής).



Εικόνα 3.2 Πως εμφανίζεται ο χάρτης και τα πεδία που υπάρχουν για κάθε παρκινγκ στη βάση δεδομένων

3.4.3 Καρτέλες των παρκινγκ

Αρχικά έγινε η χρήση μιας βιβλιοθήκης (react-native-carousel) που δημιουργούσε πρακτικά τις καρτέλες που θα βρίσκονταν τα στοιχεία των παρκινγκ, με πολύ όμορφο style και animation στην εναλλαγή των καρτελών. Παρόλα αυτά εν τέλει, δεν μπόρεσε να χρησιμοποιηθεί η βιβλιοθήκη αυτή, καθώς δεν γινόταν 'σύνδεση' με τα άλλα component της εφαρμογής, πιο συγκεκριμένα μόλις άλλαζε η καρτέλα δεν υπήρχε και αλλαγή στον τίτλο του παρκινγκ έτσι έγινε η δημιουργία εξ ολοκλήρου από την αρχή. Για να μπορεί να χρήστης να δει κάποια από αυτά τα στοιχεία των παρκινγκ, κατασκευάστηκαν κάρτες που περιέχουν το κάθε παρκινγκ ξεχωριστά. Εκτός όμως από την δημιουργία των καρτών έπρεπε να δημιουργηθεί ένα scrollview όπου θα βρίσκονται μέσα όλες οι καρτέλες και με τον συνδυασμό animation να αλλάξουν ομοιόμορφα. Επιπρόσθετα έγινε και η δημιουργία ενός καινούργιου js αρχείου όπου εάν ο χρήστης βάλει μέσα σε αυτά τα tags ένα αντικείμενο ανάλογα με την τιμή που έχει true ή false εμφανίζεται ή όχι το αντικείμενο. Στην συγκεκριμένη περίπτωση γίνεται true εάν ο χρήστης κάνει κλικ πάνω σε έναν marker. Τέλος να σημειωθεί ότι για την δημιουργία εγγραφών των παρκινγκ στη βάση δημιουργήθηκε ένα js αρχείο όπου με την χρήσης μιας εντολής γινόταν η εισαγωγή στη βάση.

Οι καρτέλες έπρεπε να μορφοποιηθούν και άλλο καθώς αποτελούν σημαντικό κομμάτι της εφαρμογής έτσι προστέθηκαν λειτουργίες όπως ανάλογα με την τιμή που έχει το παρκινγκ να εμφανίζονται κάποια σύμβολα εάν είναι φθηνό ή όχι, ένα κουμπί που δίνει τις οδηγίες για τον προορισμό (η συνάρτηση που εκτελεί την συγκεκριμένη ενέργεια έγινε αργότερα) και ένα κουμπί πληροφοριών που μεγαλώνει ακόμη περισσότερο την καρτέλα και δίνει παραπάνω πληροφορίες για το παρκινγκ. Με την αύξηση του μεγέθους της καρτέλας δημιουργήθηκε πρόβλημα στο animation των καρτελών, έτσι έπρεπε να κατασκευαστεί συνάρτηση που ανάλογα με το μέγεθος τόσο της καρτέλας όσο και της οθόνης μεταβαίνει στο επόμενο παρκινγκ. Ακολούθως έπρεπε να εισαχθεί το σύστημα rating όπου θα δίνεται η δυνατότητα βαθμολόγησης του παρκινγκ. Η εισαγωγή του έγινε από την βιβλιοθήκη react-native-elements και η χρήση της είναι πολύ εύκολη απλά εισάγοντας ένα tag rating. Το RN Elements είναι μια υλοποίηση του υλικού σχεδιασμού[10]. Το framework περιέχει ένα σύνολο συστατικών διεπαφής χρήστη γενικής χρήσης, σχεδιασμένο με παρόμοιο τρόπο. Το πολύ σημαντικό είναι ότι οι

λειτουργίες (π.χ. όταν βαθμολογεί ο χρήστης) μπορούν να αλλάξουν κατά την εκτέλεση χωρίς να κάνει rerender η εφαρμογή.

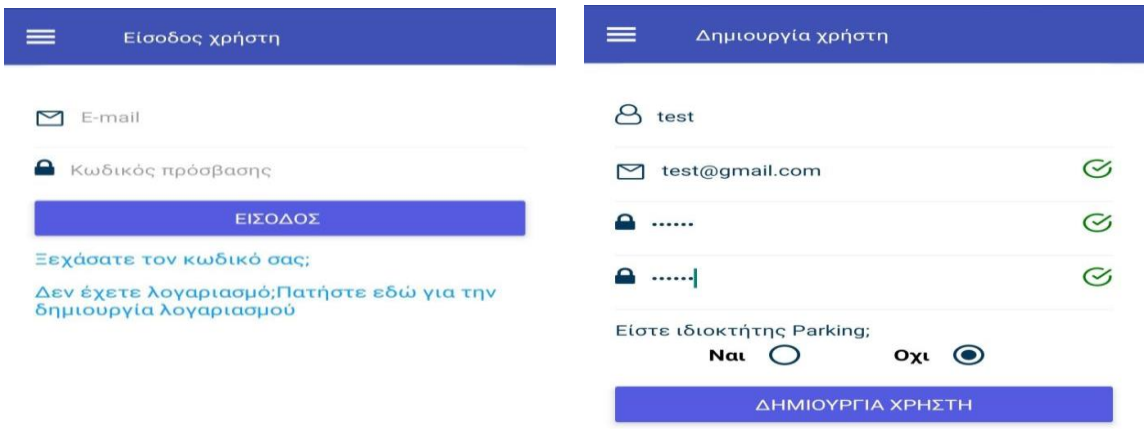
Οι τιμές που έπαιρνε το rating ερχόντουσαν από την βάση εάν είχε προστεθεί βαθμολογία. Ο χρήστης μπορεί να βαθμολογήσει το παρκινγκ μόνο εάν έχει λογαριασμό, μια δυνατότητα που δημιουργήθηκε στη συνέχεια.

3.5 Δημιουργία Authentication

Λίγα λόγια για το authentication στο firebase: Οι περισσότερες εφαρμογές πρέπει να γνωρίζουν την ταυτότητα ενός χρήστη. Η γνώση της ταυτότητας ενός χρήστη επιτρέπει σε μια εφαρμογή να αποθηκεύει με ασφάλεια τα δεδομένα του χρήστη στο cloud και να παρέχει την ίδια εξατομικευμένη εμπειρία σε όλες τις συσκευές του χρήστη. Ο έλεγχος ταυτότητας στο Firebase παρέχει υπηρεσίες backend, εύχρηστα SDK και έτοιμες βιβλιοθήκες διεπαφής χρήστη για τον έλεγχο ταυτότητας χρηστών στην εφαρμογή. Υποστηρίζει έλεγχο ταυτότητας χρησιμοποιώντας κωδικούς πρόσβασης, αριθμούς τηλεφώνου, ή άλλους τρόπους ταυτοποίησης όπως μέσω λογαριασμό Google, Facebook και Twitter και πολλά άλλα. Ο Έλεγχος ταυτότητας Firebase ενσωματώνεται και με άλλες υπηρεσίες Firebase και άλλα πρότυπα όπως το OAuth 2.0 και το OpenID Connect, ώστε να μπορεί να ενσωματωθεί εύκολα με το προσαρμοσμένο backend της εφαρμογής.

Αφού ολοκληρώθηκαν κάποια βασικά στοιχεία για την κύρια οθόνη της εφαρμογής έπρεπε να δημιουργηθούν οι χρήστες. Για τον λόγο αυτό δημιουργήθηκαν και άλλες οθόνες τόσο για την είσοδο του χρήστη όσο και για την δημιουργία. Για το register δημιουργήθηκαν κάποια πεδία που είναι υποχρεωτικά για να δημιουργήσει ο χρήστης λογαριασμό. Αυτά είναι : Όνομα χρήστη, E-mail, Κωδικός, Επιβεβαίωση κωδικού και ένα radio button που ρωτάει εάν είναι ιδιοκτήτης ή όχι παρκινγκ. Το radio Button δημιουργήθηκε σε ξεχωριστό αρχείο και απλά έγινε η εισαγωγή του στην εφαρμογή. Ουσιαστικά η λειτουργία του βασίζεται στο ότι μόλις επιλέξει ένα κύκλο ο χρήστης να μαυρίσει αυτός ο κύκλος και αν τυχόν ήταν επιλεγμένος ο άλλος κύκλος θα αποεπιλεγεί. Προφανώς για να δημιουργηθεί ο χρήστης πρέπει όλα αυτά τα πεδία να μην είναι άδεια και να πληρούν κάποιες προϋποθέσεις. Για παράδειγμα, το όνομα χρήστη και ο κωδικός πρόσβασης πρέπει να είναι πάνω από 6 χαρακτήρες, το email να είναι έγκυρο και το πεδίο επιβεβαίωση κωδικού να είναι ίδιο με τον κωδικό. Ανάλογα με το σφάλμα που υπάρχει εμφανίζεται το κατάλληλο μήνυμα (π.χ. δεν συμπλήρωσες όλα τα πεδία ή μη έγκυρο email κλπ.). Εάν όλα τα στοιχεία είναι ορθά μπορεί να δημιουργηθεί καινούργιος χρήστης. Επιπλέον οι χρήστες μπορούν να δουν εάν αυτά που έχουν εισάγει θα γίνουν δεκτά από ένα εικονίδιο που εμφανίζεται στα δεξιά κάθε text input.

Εκτός όμως από την φόρμα του authentication έπρεπε να γίνουν και ενέργειες στο firebase ώστε να αποθηκεύονται τα στοιχεία αυτά. Κατά την δημιουργία δίνονται πολλές επιλογές που μπορεί να κάνει ο χρήστης log in, εδώ επιλέχτηκε μέσω email και κωδικού πρόσβασης. Αν ο χρήστης ξεχάσει τον κωδικό του μπορεί να επιλέξει την επιλογή ξέχασα τον κωδικό μου, και να του σταλεί ένα email που του λέει να κάνει reset το password του. Αυτό γίνεται μέσω μιας αυτοματοποιημένης function που πρέπει να τοποθετηθεί το email του χρήστη και όταν καλείται στέλνει το μήνυμα (το οποίο μπορεί να προσαρμοστεί μέσω του firebase και να γίνει όπως θέλει ο προγραμματιστής) στο συγκεκριμένο email. Επίσης στο Login Screen έχει δημιουργηθεί η επιλογή να μεταφέρεται ο χρήστης στο register πατώντας στο κατάλληλο touchable opacity. Για να έχει πρόσβαση η εφαρμογή στους χρήστες θα πρέπει να ενταχθεί στο project το αρχείο Google Service-info.plist. Παράλληλα με την δημιουργία του authentication ο καινούργιος χρήστης προστίθεται και σε μία βάση στο firestore που αποθηκεύονται όλοι οι χρήστες. Αυτό συμβαίνει γιατί στοιχεία όπως εάν είναι ιδιοκτήτης παρκινγκ ή όχι αλλά και το



Εικόνα 3.3 Οι οθόνες που εμφανίζονται όταν ο χρήστης θέλει να κάνει είσοδο στην εφαρμογή και όταν θέλει να δημιουργήσει χρήστη

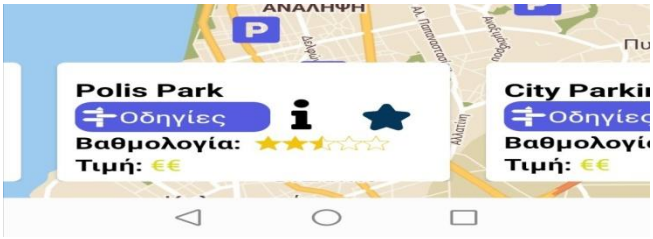
username δεν είναι εφικτό να αποθηκευτούν στο authentication, εκεί βρίσκονται μόνο το email και ο κωδικός πρόσβασης του κάθε χρήστη και το userId που δημιουργείται αυτόματα από το firebase. Εάν κάποιος χρήστης έχει πραγματοποιήσει είσοδο στην εφαρμογή και μετά αποφασίσει να την κλείσει, την επόμενη φορά που θα την ανοίξει θα είναι πάλι συνδεδεμένος στον ίδιο λογαριασμό. Τέλος, υπάρχουν κάποιες διαφορές στις καρτέλες όταν ο χρήστης είναι συνδεδεμένος που θα αναφερθούν παρακάτω.

3.6 Καρτέλες για συνδεδεμένους χρήστες

Μετά και την δημιουργία των χρηστών έπρεπε να προστεθούν οι λειτουργίες που θα ήταν ουσιώδης για να δημιουργήσει κάποιος λογαριασμό. Πρώτη λειτουργία αφορούσε το rating, στο οποίο εάν ο χρήστης δεν είναι συνδεδεμένος δεν μπορεί να πειράζει το rating που έχει η καρτέλα. Αντίθετα κάποιος ο οποίος είναι συνδεδεμένος μπορεί να ρυθμίσει το rating για οποιοδήποτε παρκινγκ και αυτό αποθηκεύεται και την επόμενη φορά που θα ανοίξει η εφαρμογή θα έχει το rating αυτό. Η λειτουργία της θα αναλυθεί στο παράρτημα, απλά πρέπει να τονιστεί ότι μόλις ο χρήστης επιλέξει ένα rating για κάποιο παρκινγκ αυτό πηγαίνει και αποθηκεύεται στο user collection στον δικό του χρήστη δημιουργώντας ένα πίνακα user Ratings όπου υπάρχουν τα παρκινγκ και οι αντίστοιχες βαθμολογίες τους. Επίσης ανάλογα εάν ο χρήστης έχει βαθμολογήσει η όχι εφαρμόζεται το αντίστοιχο rating, δηλαδή εάν έχει βάλει 4 αστέρια ένα παρκινγκ κατά την είσοδό του θα εμφανίζονται 4 αστέρια στο συγκεκριμένο ενώ εάν δεν έχει βάλει κάποιο rating θα παίρνει την default τιμή που έχει το εκάστοτε παρκινγκ. Το rating μαζί με την επόμενη λειτουργία είναι δύο εργαλεία που ειδικά για κάποιον/α που χρησιμοποιεί συνέχεια το αυτοκίνητο, είναι πολύ χρήσιμα.

Η επόμενη δυνατότητα που δίνεται στους χρήστες είναι να αποθηκεύουν τα αγαπημένα τους παρκινγκ. Με την χρήση ενός touchable opacity σε σχήμα αστεριού, ο χρήστης πατώντας πάνω αποθηκεύει το παρκινγκ αυτό στα αγαπημένα του, αλλάζει το χρώμα του αστεριού και κάθε φορά που θα κάνει log in θα βρίσκεται στα αγαπημένα του. Εάν θέλει να το αφαιρέσει από τα αγαπημένα του απλά ξανακάνει κλικ στο αστέρι και αυτό αφαιρείται. Κάνοντας κλικ ο χρήστης εκτελεί μία συνάρτηση που τοποθετεί ή αφαιρεί το συγκεκριμένο παρκινγκ που επέλεξε από το βάση δεδομένων. Επίσης εκτελούνται άλλες 2 συναρτήσεις, η μία ελέγχει ποια παρκινγκ είναι στα αγαπημένα ώστε να

τα εμφανίσει και η άλλη αλλάζει το icon του αστεριού (στο παράρτημα θα γίνει περαιτέρω ανάλυση του κώδικα).



Εικόνα 3.4 Καρτέλες παρκινγκ

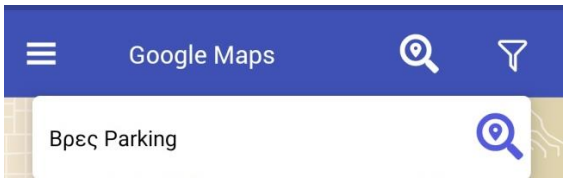
3.7 Search Bar

Το επόμενο component που δημιουργήθηκε ήταν το search bar. Η πλοήγηση και η αναζήτηση είναι δύο από τους πιο βασικούς τρόπους με τους οποίους οι επισκέπτες ανακαλύπτουν περιεχόμενο σε μία εφαρμογή. Μία μπάρα αναζήτησης αναζήτησης, μπορεί να βοηθήσει επίσης στην επίλυση τυχόν πιθανών ζητημάτων πλοήγησης. Σύμφωνα με έρευνες, εάν μία εφαρμογή προσφέρει μια δυνατότητα αναζήτησης, το 59 τοις εκατό των επισκεπτών θα το εκμεταλλευτούν. Αν μη τι άλλο, η προσθήκη μιας γραμμής αναζήτησης θα βελτιώσει την εμπειρία χρήστη. Με την τοποθέτηση search bar:

- Διευκολύνονται οι επισκέπτες να βρουν αυτό που αναζητούν, δημιουργώντας έτσι μια θετική εμπειρία χρήστη. Όταν οι επισκέπτες έχουν μια θετική εμπειρία είναι πιο πιθανό να κάνουν μια αγορά ή να δώσουν μία θετική αξιολόγηση στο playstore.

- Όταν οι επισκέπτες έχουν θετική εμπειρία χρήστη, είναι πιο πιθανό να επιστρέψουν ξανά. Η προσθήκη μιας γραμμής αναζήτησης επιτρέπει στους επισκέπτες να βρουν γρηγορότερα αυτό που αναζητούν, γεγονός που θα τους ενθαρρύνει να επιστρέψουν στο μέλλον, όταν προκύψει ανάγκη.

Επίσης η συγκεκριμένη εφαρμογή αποτελεί μία βάση με παρκινγκ οπότε θα είναι ευκολότερο για τον χρήστη να βάλει ένα όνομα ή μια τοποθεσία και να εμφανιστούν κάποια αποτελέσματα, από το να ψάχνει από μόνος του. Η δημιουργία του search bar ήταν αρκετά δύσκολη καθώς έπρεπε να συνδυαστούν πολλά πράγματα όπως όταν εμφανίζεται το αποτέλεσμα που θέλει ο χρήστης και κάνει κλικ η λίστα να φεύγει, το πληκτρολόγιο να κλείνει, να εμφανίζεται η κάρτα του παρκινγκ και να εμφανίζεται και ο τίτλος σε αυτό. Για την υλοποίηση του χρησιμοποιήθηκε ένα text input όπου κάθε φορά που πληκτρολογείτε κάτι ψάχνει εκτελείται μια function που ανάλογα με τα αποτελέσματα που κάνουν match εμφανίζεται η λίστα με αυτά. Επιπλέον μπορεί να προστεθεί η αναζήτηση να μην έγκειται μόνο στα παρκινγκ αλλά να ψάχνει και τοποθεσίες π.χ. κέντρο, καλαμαριά και να εμφανίζονται παρκινγκ από την συγκεκριμένη περιοχή. Εκτός από την λειτουργικότητα που ήταν αρκετά απαιτητική υπήρξε και πρόβλημα με το style καθώς δεν μπόρεσε να εφαρμοστεί κάτι πιο διαφορετικό. Τέλος, χρησιμοποιήθηκε και η δυνατότητα του keyboard έτσι ώστε όταν ο χρήστης πατάει πάνω στο search bar να εμφανίζεται το πληκτρολόγιο, ενώ όταν πατάει σε ένα παρκινγκ ή κάπου μέσα στον χάρτη το πληκτρολόγιο κλείνει[12].



Εικόνα 3.5 Μπάρα αναζήτησης

3.8 React-Navigation

Με την δημιουργία και του search bar, έπρεπε στην συνέχεια να δημιουργηθεί ένας τρόπος ώστε ο χρήστης να μπορεί να εναλλάσσεται μεταξύ των διαφορετικών οθονών. Εδώ μπαίνει στην εξίσωση το react-navigation (το οποίο αναλύθηκε παραπάνω), που βοηθάει στην εύκολη και γρήγορη μετάβαση από την μία οθόνη στην άλλη. Αυτή την στιγμή έχουμε στην εφαρμογή 3 βασικές οθόνες, την LoginScreen, Register Screen και την GoogleMaps Screen. Ανάλογα με την κατάσταση που βρίσκεται ο χρήστης αν είναι συνδεδεμένος ή όχι εμφανίζεται και διαφορετική οθόνη πρώτη. Αν ο χρήστης είναι συνδεδεμένος εμφανίζεται το GoogleMapScreen, αν δεν είναι συνδεδεμένος εμφανίζεται η οθόνη για να κάνει είσοδο ο χρήστης στην εφαρμογή. Όλες αυτές οι λειτουργίες Navigation έγιναν με τον Drawer Navigator δηλαδή, ο χρήστης σέρνοντας προς τα δεξιά την οθόνη εμφανίζεται η καρτέλα με τις οθόνες ή μπορεί να πατήσει στο εικονίδιο πάνω αριστερά για να εμφανιστεί. Οι λειτουργίες αυτές του Navigator πραγματοποιήθηκαν στο App.js που αποτελεί το πρώτο αρχείο που ανοίγει η εφαρμογή και ανάλογα με τις εντολές που έχει πάει στα επόμενα. Εκτός από τον Navigator έγινε και η προσθήκη του *Activity Indicator* το οποίο ουσιαστικά πρόκειται για ένα loader που εμφανίζεται μέχρι να φορτώσει η οθόνη που πρέπει να φορτωθεί. Πρόκειται για ένα πολύ χρήσιμο εργαλείο το οποίο εισάγεται απευθείας από την RN και αντί να εμφανίζει μια άσπρη οθόνη μέχρι να φορτώσει το αρχείο, δείχνει στο χρήστη ότι ουσιαστικά πρέπει να περιμένει γιατί η εφαρμογή κάνει loading κάτι. Τέλος δημιουργήθηκε ένας Navigator που εμφανίζεται μόνο όταν ο χρήστης είναι συνδεδεμένος και ουσιαστικά τον κάνει sing out από την εφαρμογή εάν θέλει να αποσυνδεθεί.

3.9 OwnerScreen

Εκτός από τους απλούς χρήστες, η εφαρμογή προσφέρει κάποιες επιπλέον δυνατότητες σε όσους είναι ιδιοκτήτες παρκινγκ. Έτσι μετά το Navigation ανάμεσα στις οθόνες δημιουργήθηκε η οθόνη για τους ιδιοκτήτες. Με την είσοδο ο χρήστης έχει δύο επιλογές, να δει τα διάφορα στατιστικά για το παρκινγκ του ή να δει κάποιες δυνατότητες που έχει για να διαφήμιση του παρκινγκ μέσα στην εφαρμογή. Αυτό οδήγησε στην δημιουργία 2 ακόμα οθονών, την Statistics.js και Ads.js. Για να είναι δυνατή η πλοήγηση μπρος και πίσω ανάμεσα σε αυτές τις 3 οθόνες, έπρεπε να δημιουργηθεί ακόμη ένας Navigator αυτή την φορά όμως *Stack* navigator καθώς δεν ήταν αναγκαία η πλοήγηση με σύρσιμο της οθόνης. Για την σωστή όμως πλοήγηση των οθονών αυτών ο *Stack* navigator μπήκε σε ένα τρίτο αρχείο ονόματι Navigator και από εκεί ουσιαστικά γίνεται η εναλλαγή των οθονών. Στην αρχική οθόνη του Owner Screen εμφανίζονται 2 μορφοποιημένα touchable opacity, το ένα οδηγεί στις διαφημίσεις και το άλλο στα στατιστικά.

3.9.1 Admob firebase

Για την χρήση των διαφημίσεων χρησιμοποιήθηκε το *admob* από το firebase. Θα πρέπει πάλι όπως για όλες τις υπηρεσίες του firebase να γίνει ενεργοποίηση του *admob* και να προστεθεί στην εφαρμογή. Κατά την εγκατάσταση της βιβλιοθήκης σημειώθηκαν αρκετά προβλήματα καθώς ήταν αδύνατη η

χρήση του, και δημιουργήθηκαν σφάλματα. Ενώ γενικά υπάρχουν τρεις παρόμοιες βιβλιοθήκες που μπορούν να χρησιμοποιηθούν, καμία τους δεν δούλευε. Για το λόγο αυτό έγιναν αρκετές αλλαγές στην εφαρμογή όπως αλλαγή στις εκδόσεις των βιβλιοθηκών, αναβάθμιση του npm, και σημαντικές αλλαγές στο build.grandle. Μετά από αυτές τις πολλές αλλαγές που έγιναν στον native κώδικα, δημιουργήθηκε το βασικό τμήμα την οθόνης που ουσιαστικά αποτελείται από από μία περιγραφή για το τι κάνει η κάθε δυνατότητα διαφήμισης και έπειτα ο χρήστης μπορεί να δει πως ακριβώς φαίνεται αυτή η διαφήμιση, πατώντας πάνω στο σύνδεσμο που υπάρχει.

3.9.2 Στατιστικά παρκινγκ

Στη δεύτερη επιλογή που έχουν οι ιδιοκτήτες, μπορούν να δουν διάφορα στατιστικά για το παρκινγκ τους. Για την δημιουργία των γραφημάτων χρησιμοποιήθηκε η βιβλιοθήκη react-native-svg και react-native-svg-charts. Για την χρήση των γραφημάτων δημιουργήθηκε ένα αρχείο με τις βασικές λειτουργίες και ένα άλλο όπου ο προγραμματιστής πρέπει να ορίσει τις τιμές που θέλει π.χ. μέγιστο, ελάχιστο ή από που θα παίρνει τα δεδομένα και απλά γίνεται η εισαγωγή του στην εφαρμογή, πρόκειται δηλαδή για ένα επαναχρησιμοποιούμενο component που μπορεί να τοποθετηθεί και σε άλλες περιπτώσεις. Μπορεί να γίνει σύγκριση γραφικών παραστάσεων μεταξύ παρκινγκ απλά αλλάζοντας τα δεδομένα που θα παίρνει ο πίνακας.

Αρχικά έγινε η δημιουργία δύο Button, το ένα είναι για να δείχνει την εβδομάδα στο γράφημα και το άλλο για να δείχνει τα αποτελέσματα για όλο τον μήνα. Ακριβώς από κάτω υπάρχει ένα text με δύο κουμπιά βελάκια δεξιά και αριστερά, όπου εάν ο χρήστης πατήσει αλλάζει ημερομηνία και κατά συνέπεια εάν υπάρχουν δεδομένα για το συγκεκριμένο παρκινγκ αυτές τις ημερομηνίες αλλάζει και το γράφημα. Για την εναλλαγή των ημερομηνιών χρησιμοποιείται η βιβλιοθήκη moment. Η MomentJs είναι μια εξαιρετικά χρήσιμη βιβλιοθήκη Javascript όταν υπάρχουν περίπλοκα πράγματα σχετικά με την ημερομηνία και την ώρα. Αναλύει επικυρώνει, χειρίζεται και εμφανίζει ημερομηνίες και ώρες σε Javascript.

Για το γράφημα χρησιμοποιήθηκαν 2 τύποι γραφήματος, με την χρήση γραμμής και με μπάρες. Οι επιλογές που δίνει η βιβλιοθήκη σχετικά με την σχεδίαση του γραφήματος είναι πάρα πολλές, οπότε ο χρήστης μπορεί να δημιουργήσει το αποτέλεσμα που θέλει. Το γράφημα χωρίζεται σε τρία μέρη τον άξονα Χ, τον άξονα Υ και το κύριο γράφημα, όπου και τα τρία μπορούν να μορφοποιηθούν ξεχωριστά. Το γράφημα δίνει μία επιλογή να εμφανίζονται ή όχι μικρές γραμμές πάνω στο άξονα χ το που μπορεί να ενεργοποιηθεί βάζοντας true ή false. Το πως δημιουργούνται θα αναλυθεί στο επόμενο κεφάλαιο. Στο Linechart η δημιουργία των κύκλων πάνω στις τιμές του γραφήματος δεν γίνεται αυτόματα, είναι συνδυασμός των 2 βιβλιοθηκών που αναφέρθηκαν παραπάνω και έτσι το μέγεθος, χρώμα ή το πόσο κοντά ή μακριά θα βρίσκονται από το σημείο του γραφήματος είναι εύκολο να οριστούν. Επίσης είχε δημιουργηθεί και η δυνατότητα όταν ο χρήστης κάνει κλικ πάνω στο κύκλο να εμφανίζεται ένα tooltip που θα λέει την τιμή που έχει αλλά λόγο του ότι τα tooltips δεν εμφανιζόντουσαν σταθερά με τις τιμές καθώς υπάρχει και θέμα με το compatibility με άλλες συσκευές, δεν χρησιμοποιήθηκε εν τέλει. Όσο αναφορά τα δεδομένα των γραφημάτων δείχνουν ουσιαστικά πόσοι χρήστες έχουν κάνει κλικ στο παρκινγκ του ιδιοκτήτη που είναι συνδεδεμένος τις συγκεκριμένες μέρες. Αυτό επιτυγχάνεται με την δημιουργία μιας συνάρτησης η οποία όποιος χρήστης είναι συνδεδεμένος, αν κάνει κλικ σε κάποιο παρκινγκ δημιουργεί για την συγκεκριμένη ημερομηνία έναν μετρητή που μετράει τα κλικ (η αντιστοίχιση μεταξύ των 2 πινάκων γίνεται μέσω του id που έχει το παρκινγκ).

Για την εναλλαγή των δεδομένων του γραφήματος όταν ο χρήστης πατάει κάποιο από τα δύο βελάκια, έπρεπε να δημιουργηθεί μία συνάρτηση. Αρχικά πριν φορτωθεί το γράφημα εμφανίζεται ένας loader επειδή θέλουν κάποια δευτερόλεπτα μέχρι να περαστούν τα δεδομένα. Η συνάρτηση βρίσκει ποιο παρκινγκ ανήκει στον χρήστη, μετά βρίσκει το παρκινγκ από την βάση δεδομένων που υπάρχουν όλα τα παρκινγκ για να πάρει το id του και μετά περνάει τις ημερομηνίες και τις τιμές σε 2 πίνακες. Ακολούθως πάει και συγκρίνει εάν οι ημερομηνίες της συγκεκριμένης εβδομάδας υπάρχουν στη βάση, εάν ναι βάζει της τιμή του counter που μετράει την επισκεψιμότητα, αλλιώς τοποθετεί 0. Έτσι εμφανίζονται τα δεδομένα για την συγκεκριμένη ημερομηνία. Εάν ο χρήστης πατήσει το βελάκι για να αλλάξει ημερομηνία, εμφανίζεται ένας δεύτερος loader μέσα στο γράφημα μέχρι να περάσουν τα δεδομένα στο γράφημα. Με αυτόν τον τρόπο ο ιδιοκτήτης μπορεί να δει για οποιαδήποτε μέρα του χρόνου, πόσοι χρήστες επισκέφτηκαν το παρκινγκ του, με την μορφή γραφικής αναπαράστασης.

Πάνω και κάτω από το γράφημα βρίσκονται 2 διακόπτες, ο ένας είναι για να αλλάζει τον τύπο του γραφήματος από linechart σε barchart και ο άλλος για να εμφανίζει στατιστικά στοιχεία των παρκινγκ. Η εναλλαγή από το ένα γράφημα στο άλλο επιτυγχάνεται την χρήση if, όπου εάν είναι ενεργό φαίνεται το ένα γράφημα ενώ μόλις ενεργοποιηθεί εμφανίζεται το άλλο γράφημα. Στο επόμενο switch όταν ενεργοποιηθεί, εμφανίζονται ο μέσος όρος, μέγιστο και το ελάχιστο από τις τιμές του γραφήματος. Επιπλέον κάτω από αυτά, μπορεί να δει ο χρήστης των αριθμό των χρηστών που έχουν προσθέσει το παρκινγκ του στα αγαπημένα τους, το οποίο επιτυγχάνεται με την βοήθεια μιας συνάρτησης. Τέλος έγινε εισαγωγή της 'οθόνης' των ιδιοκτητών στο κύριο navigator κάτι το οποίο ήταν αρκετά χρονοβόρο, καθώς έπρεπε να γίνεται έλεγχος εάν ο χρήστης που συνδέθηκε είναι ιδιοκτήτης ή όχι και ανάλογα να εμφανίζει την οθόνη των ιδιοκτητών και την δυνατότητα πρόσβασης της από τον Drawer Navigator (παρακάτω είναι οι εικόνες για τα στατιστικά και για τις διαφημίσεις).

Εικόνα 3.6 Οθόνη διαφημίσεων και στατιστικών για τους ιδιοκτήτες

3.10 Δημιουργία εικονιδίου για την εφαρμογή

Μετά από την ολοκλήρωση ενός μεγάλου μέρους της εφαρμογής, ήταν σειρά να φτιαχτεί το λογότυπο. Αρχικά έγινε μία έρευνα το πως είναι άλλα λογότυπα εφαρμογών, και προφανώς τα περισσότερα από αυτά περιέχουν το γράμμα P ώστε να γίνεται παραπομπή ότι πρόκειται για

παρκινγκ. Η συγκεκριμένη εφαρμογή όμως απευθύνεται αποκλειστικά για την πόλη της Θεσσαλονίκης, έτσι έπρεπε να δοθεί έμφαση τόσο στο να παραπέμπει σε χώρο στάθμευσης όσο και για ποια τοποθεσία πρόκειται. Για την δημιουργία του λογότυπου χρησιμοποιήθηκαν αρχικά online εργαλεία που δίνουν την δυνατότητα στο χρήστη να επεξεργάζεται διάφορα icons αλλά είχαν πολύ συγκεκριμένες δυνατότητες και επίσης δεν γινόταν merge δύο εικονιδίων. Έτσι αφού βρέθηκαν τα κατάλληλα icons χρησιμοποιήθηκε το Photoshop για καλύτερο αποτέλεσμα. Έγιναν αρκετοί πειραματισμοί όπως να υπάρχει ένα μεγάλο P με τον λευκό πύργο δίπλα αλλά δεν ήταν τόσο ελκυστικό στο μάτι. Εν τέλει, βρέθηκε η χρυσή τομή χρησιμοποιώντας ένα εικονίδιο του λευκού πύργου μαζί με ένα σύμβολο marker που παραπέμπει σε τοποθεσία. Από όλες τις δοκιμές προκρίθηκαν οι παρακάτω εικόνες και προτιμήθηκε η τελευταία. Για να προστεθεί στην εφαρμογή απλά, χρειάζεται μία αλλαγή στην θέση που βρίσκεται το default εικονίδιο των android εφαρμογών και να μπει το λογότυπο που φτιάχτηκε.

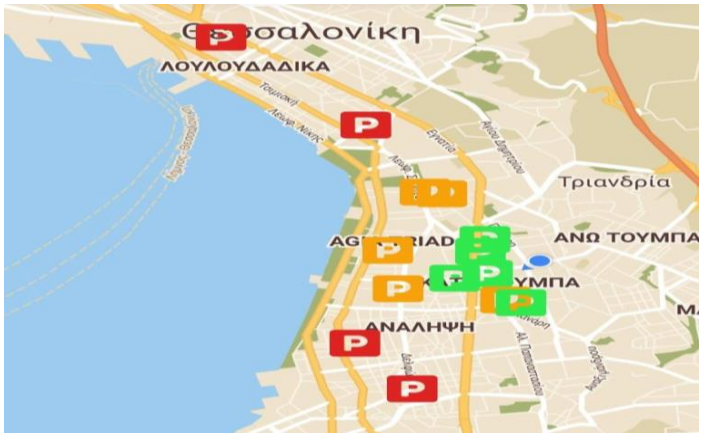
Εικόνα 3.7 Εικονίδια που δημιουργήθηκαν για την εφαρμογή

3.11 Skyline Operator

Αφού ολοκληρώθηκε και η δημιουργία του εικονιδίου, υλοποιήθηκε μία πολύ βασική λειτουργία της εφαρμογής, που είναι η κατηγοριοποίηση των παρκινγκ. Η συνάρτηση βασίστηκε στον skyline operator που αναφέρθηκε σε προηγούμενο κεφάλαιο. Η κατηγοριοποίηση ενεργοποιείται μόνο όταν ο χρήστης βρίσκεται στην οθόνη Google Map Screen όπου βρίσκεται ο χάρτης με τα παρκινγκ. Εδώ να σημειωθεί πως αρχικά έγινε προσπάθεια ενσωμάτωσης του switch στο Navigation, δηλαδή κάτω από το μέρος που γίνεται η περιήγηση στις οθόνες της εφαρμογής (Register, Login κ.τ.λ.). Ενώ μπόρεσε να προστεθεί όπως και να εμφανίζεται όταν η οθόνη GoogleMaps είναι Focus, όταν όμως άλλαζε η κατάσταση του διακόπτη από enable disable και το αντίθετο δεν γινόταν εκτέλεση της συνάρτησης, όποτε εγκαταλείφτηκε αυτή η ιδέα.

Πατώντας πάνω δεξιά στο εικονίδιο των φίλτρων, εμφανίζεται ένα scroll View όπου βρίσκεται και ο διακόπτης που ενεργοποιεί την συνάρτηση. Αρχικά γίνεται ο υπολογισμός της τοποθεσίας του χρήστη, το οποίο υπολογίζεται με την βοήθεια της βιβλιοθήκης react-native-location. Αφού πάρει τα latitude και longitude του χρήστη, με την χρήση μιας συνάρτησης παίρνει την απόσταση από το κάθε παρκινγκ και τα περνάει σε ένα πίνακα. Μέχρι να γεμίσει με 10 εγγραφές ο πίνακας, όλα τα παρκινγκ μπαίνουν κανονικά, στη συνέχεια όμως με την χρήση της function get Array Max αποθηκεύεται η μεγαλύτερη απόσταση σε μία μεταβλητή και σε κάθε επανάληψη μέχρι να τελειώσουν όλες οι εγγραφές, συγκρίνει εάν αυτή η απόσταση είναι μεγαλύτερη σε σχέση με μία άλλη και αν ναι γίνεται αντικατάσταση της. Η διαδικασία αυτή επαναλαμβάνεται μέχρι να μείνουν τα 10 παρκινγκ με την μικρότερη απόσταση από τον χρήστη.

Αφού δημιουργηθεί ο πίνακας με τις μικρότερες αποστάσεις, χωρίζεται σε 2 μέρη, τα οποία αποθηκεύονται σε διαφορετικούς πίνακες. Το πρώτο μέρος είναι τα παρκινγκ με την μικρότερη απόσταση με τον χρήστη που θεωρούνται βέλτιστα και έχουν πράσινο χρώμα. Σε περίπτωση που 2 παρκινγκ απέχουν ίδια απόσταση από τον χρήστη (γίνεται στρογγυλοποίηση των αποστάσεων ώστε να μην είναι ο μόνος παράγοντας στο κριτήριο της κατηγοριοποίησης), συγκρίνονται ανάλογα με την τιμή τους, και όποιο έχει την μικρότερη αυτό τοποθετείται στον βέλτιστο πίνακα. Η ίδια διαδικασία επαναλαμβάνεται με τον αντίστροφο τρόπο για τα υπόλοιπα. Δηλαδή επιλέγονται αυτά με την μεγαλύτερη απόσταση και αν η απόσταση είναι ίδια επιλέγονται αυτά με την μεγαλύτερη τιμή. Όλα τα υπόλοιπα χρωματίζονται με κόκκινο χρώμα. Τέλος η συνάρτηση εκτελείται και όταν γίνει το πρώτο render της εφαρμογής ώστε να προσδιοριστούν οι αποστάσεις του χρήστη από τα παρκινγκ, και ξανά εκτελείται όταν ενεργοποιηθεί το διακόπτης.



Εικόνα 3.8 Αποτέλεσμα όταν εφαρμόζεται ο αλγόριθμος βελτιστοποίησης

3.12 Δημιουργία Directions

Μία βασική λειτουργία του ThessParking, είναι να δίνει οδηγίες στο χρήστη για να φτάσει στον προορισμό του. Η λειτουργία αυτή ενεργοποιείται μόλις ο χρήστης κάνει κλικ στο κουμπί οδηγίες που βρίσκεται πάνω σε κάθε καρτέλα κάθε παρκινγκ. Για να μπορέσει ο χρήστης να δει την διαδρομή που πρέπει να ακολουθήσει, πρέπει να δοθεί από την direction api. Στην συγκεκριμένη περίπτωση χρησιμοποιήθηκε το Directions Api από την google το οποίο είναι δωρεάν για έναν μεγάλο αριθμό calls. Έπειτα, δημιουργήθηκε μία συνάρτηση που εκτελείται όταν ο χρήστης κάνει κλικ στο κουμπί οδηγίες και αν δεν είναι ενεργοποιημένο το gps εμφανίζει μήνυμα που προτρέπει τον χρήστη να το ανοίξει. Μόλις είναι ενεργοποιημένο, εκτελείται η function getDirections, που υπολογίζει την τοποθεσία του χρήστη, του παρκινγκ, κάνει focus στον χρήστη ώστε να ακολουθεί την διαδρομή του.

Εικόνα 3.9 Αποτέλεσμα όταν πατηθεί το κουμπί οδηγίες

Για να κατασκευαστεί η γραμμή της διαδρομής χρησιμοποιήθηκε η βιβλιοθήκη react-native-maps-directions, όπου με τα στοιχεία από την προηγούμενη συνάρτηση σχηματίζει την διαδρομή. Τέλος μόλις ενεργοποιείται η διαδρομή πάνω δεξιά από τις καρτέλες εμφανίζεται ένα κόκκινο κουμπί κλεισίματος το οποίο μόλις πατηθεί σταματάει την διαδρομή.

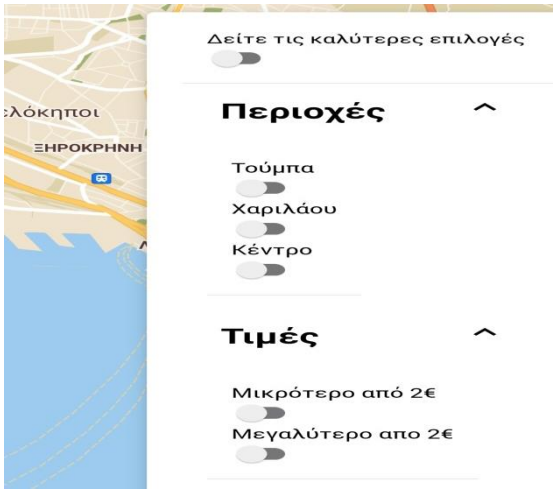
3.13 Δημιουργία καρτέλας αγαπημένων

Όταν ο χρήστης είναι συνδεδεμένος έχει την δυνατότητα να αποθηκεύει τα αγαπημένα του παρκινγκ. Για το λόγο δημιουργήθηκε ακόμη μία καρτέλα που δείχνει τα παρκινγκ αυτά, και εμφανίζεται όταν ο χρήστης κάνει log in. Μέχρι τα περάσουν τα δεδομένα στον πίνακα όπου τα δείχνει στην οθόνη, εμφανίζεται ένας loader σαν τους προηγούμενους. Μετά φτιάχτηκε μία συνάρτηση όπου βρίσκει όλα τα αγαπημένα του, από την βάση, και τα αποθηκεύει σε έναν πίνακα και έπειτα καλείται μία άλλη συνάρτηση που βρίσκεται μέσα σε ένα scrollview για να δείξει όλα τα δεδομένα. Στα πεδία τιμή και rating χρησιμοποιήθηκαν οι συναρτήσεις που υπάρχουν στις καρτέλες για κάθε παρκινγκ στο αρχείο που δείχνει τον χάρτη, για να εμφανίζονται με αυτό τον τρόπο όπως φαίνονται παρακάτω.

Εικόνα 3.10 Καρτέλα αγαπημένα παρκινγκ

3.14 Δημιουργία φίλτρων

Στο εικονίδιο των φίλτρων που εμφανίζει στον χρήστη τα καλύτερα για αυτόν παρκινγκ, δημιουργήθηκαν και άλλα 2 φίλτρα. Η μία κατηγορία φίλτρων είναι οι τοποθεσίες και η άλλη οι τιμές. Οι επιλογές των δύο κατηγοριών εμφανίζονται όταν ο χρήστης πατήσει πάνω τους και εμφανίζονται οι διακόπτες. Η λειτουργία αυτή δημιουργήθηκε με το component Accordian, το οποίο δέχεται 2 τιμές τον τίτλο και τα στοιχεία που θα εμφανίσει και είναι ουσιαστικά ένα touchableopacity που όταν πατηθεί μεγαλώνει δείχνει τα στοιχεία που περιέχει. Κάθε διακόπτης έχει ξεχωριστό state (true,false) και ξεχωριστή συνάρτηση που ενεργοποιείται, κάθε φορά που ο χρήστης κάνει κλικ. Οι συναρτήσεις αυτές είναι παρόμοιες μεταξύ τους απλά κάθε φορά αλλάζουν οι τιμές ανάλογα με το ποιος διακόπτης ενεργοποιείται. Οι διακόπτες των περιοχών μπορούν να ενεργοποιηθούν ταυτόχρονα, αλλά απενεργοποιούνται όταν ενεργοποιηθεί ένας διακόπτης από τις τιμές (οι διακόπτες των τιμών δεν μπορούν να είναι ενεργοί ταυτόχρονα).

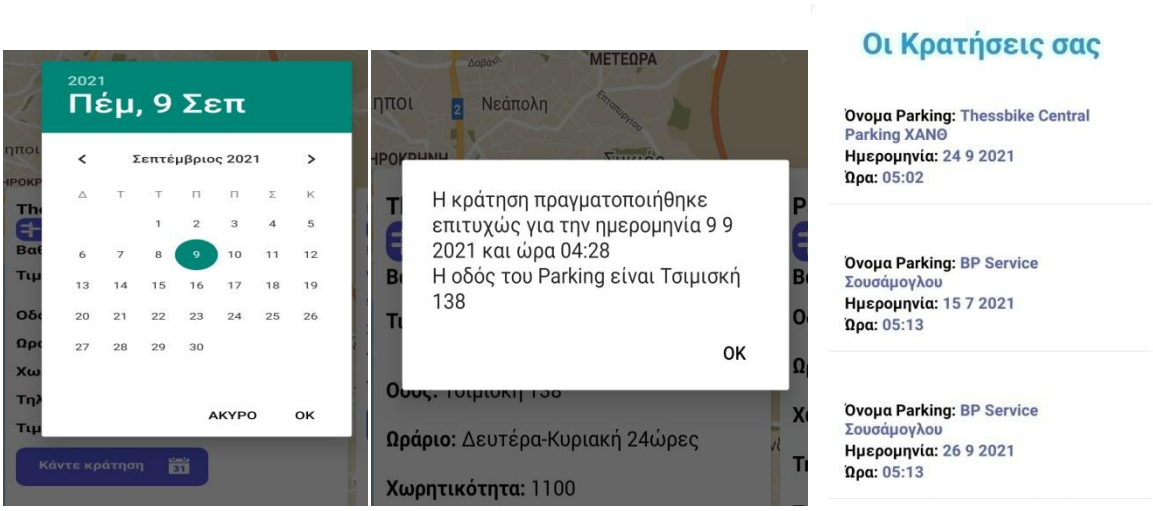


Εικόνα 3.11 Καρτέλα που εμφανίζονται τα φίλτρα

3.15 Δημιουργία κρατήσεων

Τελευταία δυνατότητα που προστέθηκε ήταν η δυνατότητα να κάνει κράτηση ο χρήστης. Η δημιουργία του κουμπιού που δίνει αυτή την δυνατότητα έγινε σε ξεχωριστό αρχείο και απλά έγινε εισαγωγή του component στην καρτέλα κάθε παρκινγκ. Όταν ο χρήστης κάνει κλικ πάνω στο κουμπί ελέγχεται αν είναι συνδεδεμένος ή όχι. Αν δεν είναι εμφανίζετε αντίστοιχο μήνυμα που τον προτρέπει να συνδεθεί και αν είναι συνδεδεμένος του εμφανίζεται πρώτα ένα ημερολόγιο για να επιλέξει ημερομηνία και έπειτα να επιλέξει ώρα, εάν ήταν επιτυχής η κράτηση του εμφανίζει αντίστοιχο μήνυμα. Για την δημιουργία του ημερολογίου χρησιμοποιήθηκε η βιβλιοθήκη @react-native-community/datetimepicker, όπου δημιουργεί ένα calendar ή ένα ρολόι και σε όποιες τιμές ο χρήστης κάνει κλικ αυτές μπορούν να αποθηκευτούν σε κάποιες μεταβλητές και να χρησιμοποιηθούν. Έτσι εάν η κράτηση είναι επιτυχής αποθηκεύονται στη βάση ποιος χρήστης έκανε την κράτηση, πότε και σε ποιο παρκινγκ. Επιπλέον δημιουργήθηκε και μία καρτέλα για τις κρατήσεις, παρόμοια με εκείνη

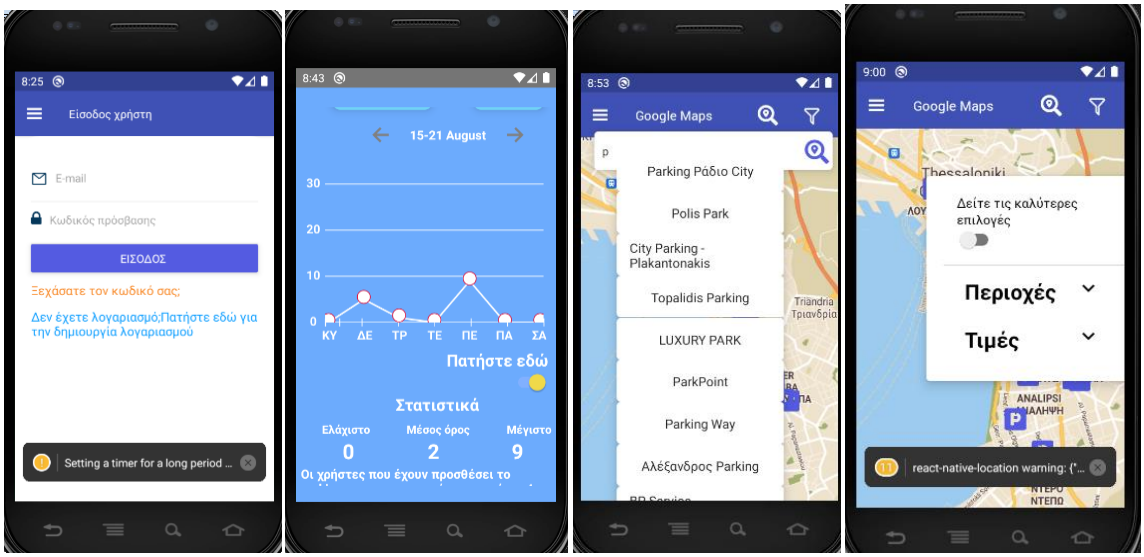
για τα αγαπημένα παρκινγκ. Για την εμφάνιση των σωστών δεδομένων απλά συγκρίνονται ποιες κρατήσεις έχουν το ίδιο email με τον συνδεδεμένο χρήστη και εμφανίζονται.



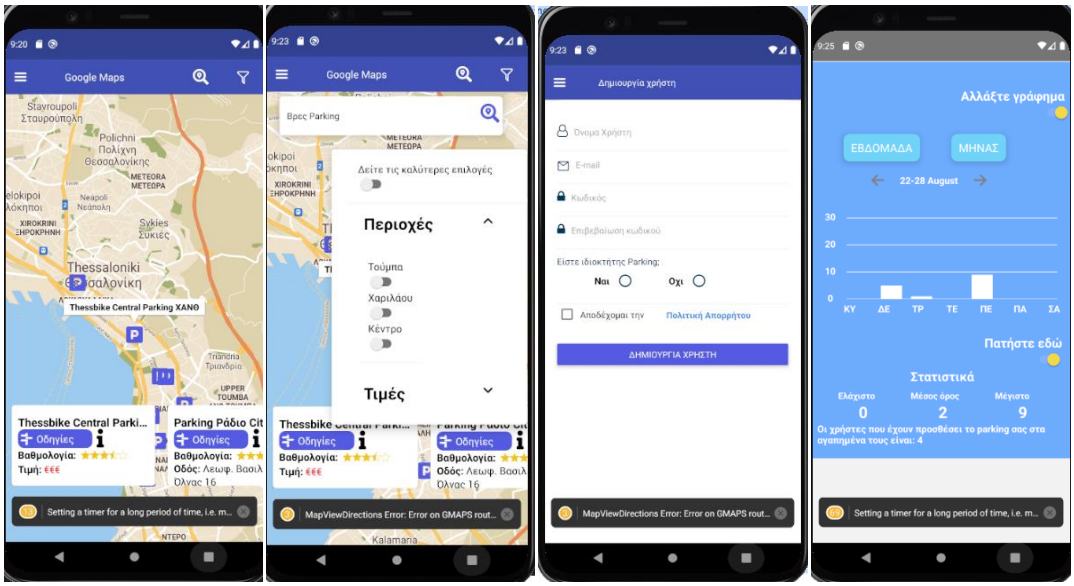
Εικόνα 3.12 Εμφάνιση ημερολογίου και καρτέλας κρατήσεων

3.16 Συμβατότητα με άλλες συσκευές

Αφού ολοκληρώθηκε ο προγραμματισμός της εφαρμογής έπρεπε να γίνει έλεγχος πως φαίνεται η εφαρμογή σε συσκευές διαφορετικού μεγέθους. Επιλέχτηκαν 2 συσκευές, το Pixel 4 XL που αποτελεί μία μεγάλη συσκευή, και το nexus S που είναι μία μικρή συσκευή(6.3 ίντσες και 4 αντίστοιχα). Δημιουργήθηκαν έτσι οι emulators για το κάθε τηλέφωνο, και εξετάσθηκαν τα σημεία που χρειαζόντουσαν επιδιόρθωση. Ευτυχώς σε ελάχιστα σημεία υπήρχε πρόβλημα με τα styles των components και αυτό διότι τα περισσότερα μεγέθη είναι με ποσοστά ανάλογα με το μέγεθος της οθόνης. Παρακάτω υπάρχουν στιγμιότυπα και από τις δύο συσκευές.



Εικόνα 3.13 Στιγμιότυπα από Nexus s



Εικόνα 3.14 Στιγμιότυπα από Pixel 4 XL

3.17 Επίλογος

Η διαδικασία ολοκλήρωσης της εφαρμογής όπως ήταν αναμενόμενο αποτέλεσε μία χρονοβόρα διαδικασία. Ιδιαίτερα σε αρχικά στάδια, αφού κάποιες βιβλιοθήκες που έπρεπε να χρησιμοποιηθούν, ήθελαν ιδιαίτερη μεταχείριση στην εγκατάσταση τους. Ακόμα εκτός από τα νέα εργαλεία της RN που χρησιμοποιήθηκαν πρώτη φορά, έπρεπε να γίνει εξοικείωση και με το firebase. Επίσης η εφαρμογή περιέχει πολλές λειτουργίες και όλες δημιουργήθηκαν από την αρχή (βαθμολόγηση, αγαπημένα, directions,στατιστικά κ.α.). Παρόλα αυτά, στο τελευταίο μέρος υλοποίησης της εφαρμογής, χρησιμοποιήθηκαν components που είχαν ήδη φτιαχτεί άρα δεν χάθηκε πολύς χρόνος. Επιπρόσθετα ρόλο σε αυτό έπαιξε η εξοικείωση με την εφαρμογή, αφού η καθημερινή επαφή είναι ο καλύτερος τρόπος για να γίνει μία γλώσσα προγραμματισμού πιο κατανοητή στο χρήστη.

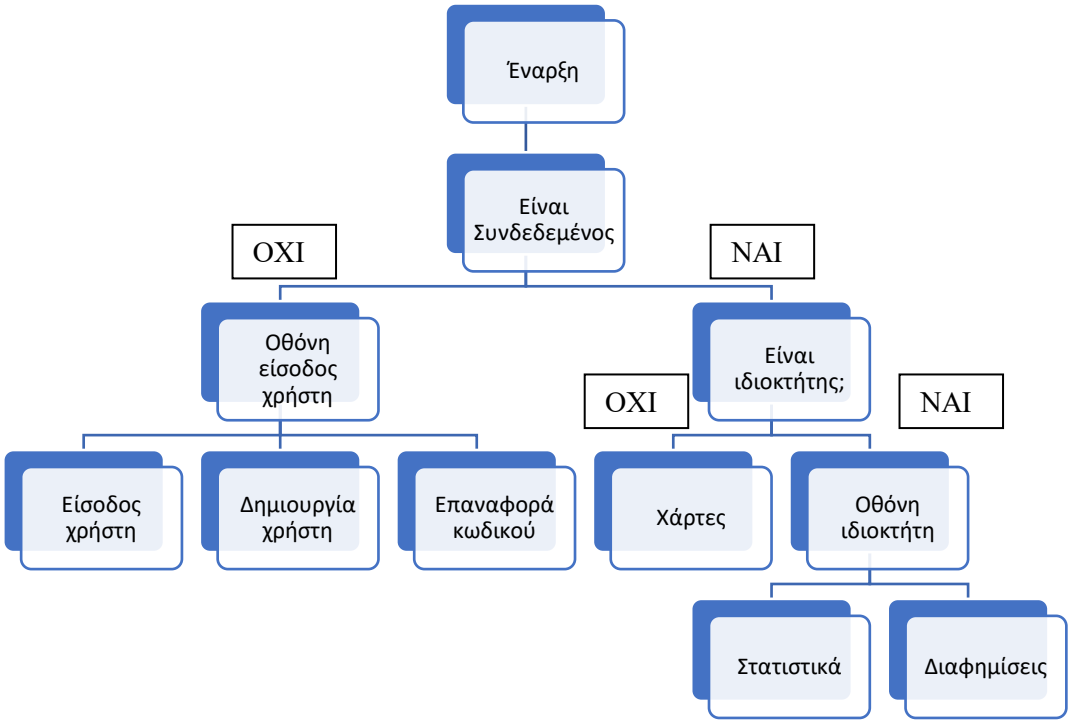
Κεφάλαιο 4ο: Δομή προγράμματος

4.1 Εισαγωγή

Η δομή ενός προγράμματος παίζει πολύ σημαντικό ρόλο στην λειτουργία του. Παρακάτω θα αναλυθούν ο τρόπος λειτουργίας του προγράμματος καθώς και τι σκοπό έχει κάθε component και οθόνη στο πρόγραμμά.

4.2 Navigation

Το Navigation είναι ο κύριος ρυθμιστής του προγράμματος και αποφασίζει ποια οθόνη θα εμφανιστεί ανάλογα με τα credentials που έχει ο χρήστης της εφαρμογής. Εάν κάποιος συνδεθεί χωρίς να έχει κάνει προηγουμένως log in θα τον εμφανίσει στην φόρμα για να συνδεθεί στο λογαριασμό του (εάν έχει). Αν ανοίξει την εφαρμογή και είναι συνδεδεμένος και είναι ιδιοκτήτης θα του εμφανίσει την καρτέλα που εμφανίζεται μόνο στους ιδιοκτήτες, ενώ αν δεν είναι, θα τον βγάλει απευθείας στο google maps. Εκτός από την οθόνη των ιδιοκτητών όλες οι υπόλοιπες οθόνες είναι προσβάσιμες από όλους τους χρήστες ανεξαρτήτως σε ποια οθόνη βρίσκονται. Ο χρήστης μπορεί να επιστρέψει στην προηγούμενη οθόνη πατώντας το πίσω κουμπί.



Σχήμα 4.1 Διάγραμμα Προγράμματος

4.3 Είσοδος-Δημιουργία χρήστη

4.3.1 Δομή οθόνης εισόδου χρήστη

Η δομή του Login αποτελείται από το render και 2 σημαντικές function. Στο render βρίσκεται η φόρμα συμπλήρωσης στοιχείων, που αν είναι σωστά ο χρήστης κάνει log in. Οι άλλες 2 συναρτήσεις ενεργοποιούνται όταν ο χρήστης πατήσει πάνω στα touchable opacities που υπάρχουν και μεταφέρεται είτε στην επαναφορά κωδικού είτε στην δημιουργία χρήστη.

4.3.2 Δομή οθόνης δημιουργίας χρήστη

Το Register Screen είναι παρόμοιο με το log in με την φόρμα να βρίσκεται στο render αλλά να περιέχει περισσότερες function για ελέγχους. Ελέγχει το όνομα χρήστη, τον κωδικό, το email, την επανάληψη κωδικού και αν όλα πληρούν τις προϋποθέσεις δημιουργεί χρήστη αλλιώς, εμφανίζει που βρίσκεται το σφάλμα. Επίσης περιέχει πολλά state καθώς για κάθε πεδίο εάν είναι σωστό εμφανίζεται ένα εικονίδιο δίπλα, οπότε έπρεπε για κάθε πεδίο να γίνει ξεχωριστό state.

4.4 Οθόνη ιδιοκτητών

Όταν ο χρήστης είναι ιδιοκτήτης εμφανίζεται η οθόνη Owner Screen που είναι πρακτικά ένας navigator για τα στατιστικά και τις διαφημίσεις. Η δομή του είναι πολύ απλή, αφού αποτελείται από 2 συναρτήσεις που μεταφέρουν τον χρήστη στην αντίστοιχη οθόνη, οι οποίες εκτελούνται όταν πατήσει σε ένα από τα δύο touchable opacities που υπάρχουν.

4.4.1 Στατιστικά

Εμφανίζεται όταν ο χρήστης κάνει κλικ στα στατιστικά, και η δομή της δεν είναι πολύπλοκη γιατί είναι ουσιαστικά ένα component το οποίο έχει δημιουργηθεί σε άλλο αρχείο, και εδώ μπαίνουν μόνο οι τιμές (data,color ymin, ymax κ.τ.λ.). Η δομή του component που βασίζεται, σε αντίθεση με τα περισσότερα είναι functional περιέχει δηλαδή μόνο συναρτήσεις και μία ‘κύρια’ συνάρτηση όπου εκτελούνται οι περισσότερες από αυτές.

- Τέσσερις συναρτήσεις σχετίζονται με την βιβλιοθήκη moment για να αλλάζει η ημερομηνία στην οθόνη
- Μία που αλλάζει τον τύπο του γραφήματος, μία άλλη που εμφανίζει τα κυκλάκια στο Line Chart
- Την function που ελέγχει πόσοι χρήστες έχουν το συγκεκριμένο παρκινγκ στα αγαπημένα τους
- Συναρτήσεις για Max και Min
- Πολλά useState γιατί αλλάζουν πολλές καταστάσεις τιμές

Το return της βασικής συνάρτησης περιέχει τα buttons που βρίσκονται στο πάνω μέρος, τα 2 switch που αλλάζουν το γράφημα και δείχνουν στατιστικά στοιχεία, το text που αλλάζει τις ημερομηνίες από τα κουμπιά και φυσικά το γράφημα που εμφανίζεται.

4.4.2 Οθόνη διαφημίσεων

Η υλοποίηση του ήταν γρήγορη και απλή αφού αποτελείται από τρία μεγάλα texts και τρεις συναρτήσεις που η κάθε μία εμφανίζει διαφορετικό τύπο διαφήμισης. Οι τύποι διαφημίσεων απλά γίνονται import από την βιβλιοθήκη react-native-firebase/admob και οι συναρτήσεις εκτελούνται μόλις ο χρήστης πατήσει πάνω στα touchable opacities.

4.5 Components

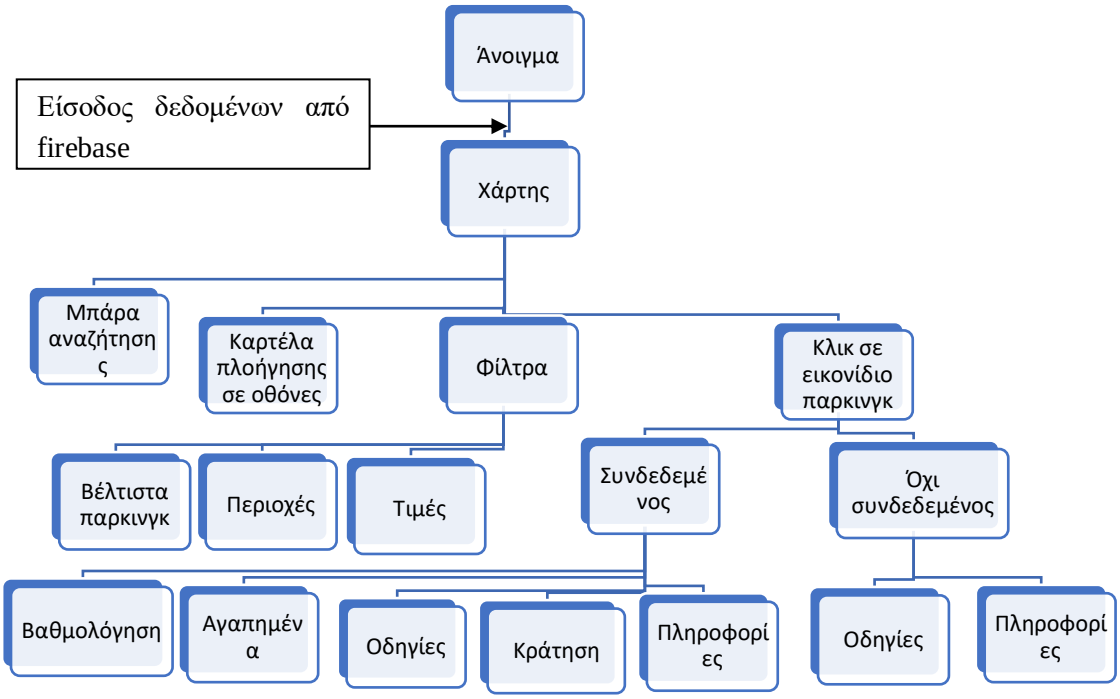
Τα components που δημιουργήθηκαν δεν ήτανε πολλά, θα μπορούσαν να ήταν σίγουρα περισσότερα αλλά πολλές φορές υπήρχε πρόβλημα στην εμφάνιση ή στις μεταβλητές που έπρεπε να περάσουν.

- Το Loading είναι ένα component που εμφανίζεται στην αρχή πριν κάνει load η εφαρμογή. Η δομή του είναι απλή αφού στο return εμφανίζεται ένας Activity Indicator (κύκλος loading), και άμα προκύψει κάποιο σφάλμα στον Navigator ίσως να μπορέσει να κάνει navigate στην κατάλληλη οθόνη.
- Το Radio Button χρησιμοποιείται μόνο στη δημιουργία χρήστη, και ουσιαστικά είναι δύο touchable opacities που όταν το ένα ενεργοποιείται το άλλο απενεργοποιείται.
- Το Myview που χρησιμοποιήθηκε πολλές φορές που αποκρύπτει και εμφανίζει στα στοιχεία που βρίσκονται μέσα στο View ανάλογα αν είναι true false η τιμή. Είναι ένα functional component που αν είναι true επιστρεφεί null ενώ true όλα τα children.
- Accordian που χρησιμοποιείται από τις κεφαλίδες των φίλτρων για να εμφανίσουν τους διακόπτες. Ουσιαστικά εμφανίζει και αποκρύπτει ένα scrollview
- DatePicker χρησιμοποιείται για να κάνει κράτηση ο χρήστης σε οποιοδήποτε παρκινγκ

4.6 GoogleMap Screen

Η οθόνη αυτή αποτελεί την κύρια οθόνη της εφαρμογής και η δομή της είναι αρκετά περίπλοκη. Αρχικά είναι ένα class based component με πολλές συναρτήσεις, πολλά state μεταβλητών και πολλές λειτουργίες ακόμα και στο return. Στην αρχή πριν την έναρξη του component, δηλώνονται κάποιες μεταβλητές που έχουν σχέση με την τοποθεσία του χάρτη αλλά και την πρόσβαση στο firebase. Έπειτα υπάρχει ο constructor με όλα τα state και τις αρχικές τους τιμές και ακριβώς από κάτω τρεις παρόμοιες συναρτήσεις που παίρνουν δεδομένα από το firebase και τα περνάνε σε arrays για να μπορούν να χρησιμοποιηθούν πιο εύκολα στην εφαρμογή. Συνεχίζοντας υπάρχουν συναρτήσεις που βοηθούν στον υπολογισμό αποστάσεων και στην κατηγοριοποίηση των παρκινγκ. Έπειτα βρίσκονται τα componentDidmount και update όπου κάνουν κυρίως ελέγχους. Ακριβώς από κάτω είναι η συνάρτηση markerclick που ενεργοποιείται μόλις ο χρήστης κάνει κλικ σε ένα παρκινγκ, και μέσω αυτής ξεκλειδώνονται ακόμα περισσότερες functions όπως η εμφάνιση των καρτελών, animations, rating, directions κ.α.

Μετά από όλες αυτές τις συναρτήσεις υπάρχει το render όπου ανάλογα με τα state των μεταβλητών εκτελεί ή όχι κάποιες function. Η δομή του return χωρίζεται στο header και στον υπόλοιπο χάρτη. Εκεί βρίσκεται το κουμπί που εμφανίζει στο χρήστη το navigation αλλά και τα φίλτρα που μπορεί να κάνει ο χρήστης κατηγοριοποίηση των χώρων στάθμευσης. Ο χάρτης ουσιαστικά δημιουργείται από το tag του mapView που υπάρχει όπου μπαίνουν συντεταγμένες και πολλές άλλες ρυθμίσεις. Τα markers των παρκινγκ εμφανίζονται τρέχοντας την function .map σε πίνακα που περιέχει τα παρκινγκ. Μετά ανάλογα αν η κατηγοριοποίηση είναι ενεργοποιημένη ή όχι χρωματίζει τα εικονίδια. Υπάρχει ακόμα το animation που εμφανίζει τις κάρτες όπως και το search bar που είναι ουσιαστικά ένα text input. Τέλος υπάρχει ένα scrollview που εμφανίζεται μόνο όταν ο χρήστης πατήσει πάνω στα φίλτρα.



Σχήμα 4.2 Διάγραμμα GoogleMapScreen

4.7 Επίλογος

Η δομή ενός προγράμματος αποτελεί ένα πολύ σημαντικό στοιχείο για την σωστή λειτουργία του. Η εφαρμογή ThessParking αποτελείται από τρεις με τέσσερις βασικές οθόνες, που είναι συνδεδεμένες μεταξύ τους, η δομή τους είναι απλή και χρησιμοποιούνε για κάποιες λειτουργίες τους components. Η εξαίρεση στον κανόνα είναι η οθόνη για τους χάρτες όπου η δομή της, είναι πολύ περίπλοκη καθώς περιέχει πολλές λειτουργίες και μεταβλητές, που ίσως κάνουν την κατανόηση του κώδικα από τρίτους δύσκολη.

Κεφάλαιο 5ο: Συμπεράσματα και προτάσεις βελτίωσης

5.1 Εισαγωγή

Στο τελευταίο κεφάλαιο θα παρατεθούν προτάσεις για βελτίωση της εφαρμογής καθώς και του προβλήματος παρκινγκ γενικότερα. Είναι πολύ σημαντικό μετά την περάτωση μιας εφαρμογής ή την ολοκλήρωση ενός έργου γενικότερα, να γίνεται μία σύνοψη το αν πετυχαίνει το σκοπό για τον οποίο δημιουργήθηκε και να παρουσιάζονται μελλοντικά σχέδια που θα μπορούσαν να βοηθήσουν στην περαιτέρω ανάπτυξη.

5.2 Βελτιώσεις στον κώδικα

Σε οποιαδήποτε κώδικα υπάρχουν πάντα δυνατότητες να γίνει πιο σύντομος ή να κάνει κάποιες λειτουργίες χωρίς να χρησιμοποιεί τόση υπολογιστική ισχύ. Στην συγκεκριμένη εφαρμογή, ο κώδικας είναι σωστά δομημένος με στοιχήσεις, σχόλια όπου χρειάζεται και όσα δυνατόν λιγότερα σφάλματα. Παρόλα αυτά, ιδιαίτερα στην κεντρική οθόνη των χαρτών θα μπορούσαν να χρησιμοποιηθούν περισσότερα components και να μην δημιουργούνται όλα μέσα στο συγκεκριμένο αρχείο. Ακόμα functions που υπάρχουν ήδη θα μπορούσαν με λιγότερο κώδικα να κάνουν τις ίδιες λειτουργίες. Επιπρόσθετα, σε μελλοντικό χρόνο θα μπορούσε να αλλαχτεί η δομή των περισσότερων JS αρχείων και από class component να γίνουν function, όταν θα χρησιμοποιούνται ευρέως τα react hooks. Αποτελεί βέβαια μία χρονοβόρα διαδικασία η αλλαγή αυτή αλλά ίσως βοηθήσει την εφαρμογή να γίνει γρηγορότερη.

5.3 Βελτιώσεις στην εφαρμογή

Το ThessParking περιέχει αρκετές δυνατότητες για τον χρήστη παρόλα αυτά, υπάρχουν πολλές ακόμα λειτουργίες που θα μπορούσαν να προστεθούν. Μία πολύ σημαντική δυνατότητα θα ήταν ο χρήστης να μπορεί να κάνει κράτηση σε όποιο παρκινγκ θέλει αλλά με την δυνατότητα να στέλνετε η ενημέρωση αυτή στο παρκινγκ που έχει κάνει κράτηση ότι την συγκεκριμένη ημερομηνία και ώρα θα έρθει ο χρήστης. Μία ακόμη λειτουργία είναι να εμφανίζονται πόσα διαθέσιμα παρκινγκ υπάρχουν, αυτό είναι κάτι που ίσως μπορέσει να επιτευχθεί εάν στο εκάστοτε παρκινγκ υπάρχει σύστημα που ελέγχει τα αμάξια που μπαίνουν και βγαίνουν και ανανεώνει μία βάση, άρα έχοντας πρόσβαση σε αυτή την βάση είναι εύκολο να εμφανιστούν τα διαθέσιμα παρκινγκ. Θα μπορούσε επίσης να προστεθούν φωτογραφίες σε κάθε χώρο στάθμευσης έτσι ώστε ο προσανατολισμός του χρήστη να γίνεται πιο εύκολα όταν βρίσκεται στο δρόμο

Για τους χρήστες που έχουν λογαριασμό και άλλες λειτουργίες, εκτός από την δυνατότητα της βαθμολογίας και να αποθηκεύουν τα αγαπημένα τους παρκινγκ. Θα μπορούσαν να αφήνουν σχόλια κάτω από το καθένα ώστε να γράφουν αναλυτικά για την εμπειρία τους και να ανταλλάσσουν απόψεις με άλλους χρήστες. Επιπλέον αφού έχουν επισκεφτεί ένα θα μπορούσε η εφαρμογή να ρωτάει τον χρήστη πόσο ικανοποιημένος έμεινε για την εξυπηρέτηση, τον χώρο, τις τιμές κ.α. να είναι δηλαδή η αξιολόγηση πιο αναλυτική. Εκτός από τα παραπάνω, θα μπορούσαν να έχουν πιο ενεργό ρόλο στην βελτίωση της εφαρμογής π.χ. Εάν ένα παρκινγκ έχει αλλάξει τις τιμές του ή δεν υπάρχει πλέον ενώ φαίνεται στην εφαρμογή, θα μπορούσε να ενημερώσει με ένα email και να κερδίζει κάποιους πόντους,

που αν μαζέψει έναν αριθμό θα έχει έκπτωση σε κάποιο παρκινγκ ή ακόμα σε συνδυασμό με αυτό που αναφέρθηκε παραπάνω, να στέλνει κάποιες φωτογραφίες και να κερδίζει πόντους.

5.4 Αντιμετώπιση της εύρεσης παρκινγκ

Η πόλη της Θεσσαλονίκης έρχεται αντιμέτωπη καθημερινά με το πρόβλημα του παρκινγκ, ιδιαίτερα στο κέντρο της πόλης. Θα πρέπει να δημιουργηθούν κτήρια αποκλειστικά για τον σκοπό αυτό, ώστε να απελευθερωθούν τα πεζοδρόμια από τα αυτοκίνητα και όποιοι παρκάρουν παράνομα να πληρώνουν χρηματικό αντίτιμο. Επίσης η δημιουργία περισσότερων μέσων συγκοινωνίας καθώς το μόνο μέσο κυκλοφορίας μαζικής μεταφοράς είναι τα λεωφορεία. Η δημιουργία του μετρό ίσως βοηθήσει στην αντιμετώπιση του προβλήματος αυτού. Επιπλέον χρήσιμο θα ήταν η δημιουργία μεγάλου συστήματος με ποδηλατοδρόμους ώστε να παρακινούν τους κατοίκους να χρησιμοποιούν μέσα που δεν είναι επιβλαβή για το περιβάλλον, οι αποστάσεις στην πόλη δεν είναι μεγάλες οπότε η χρήση ποδηλάτου θα μπορούσε να λύσει πολλά προβλήματα.

Στον τεχνολογικό τομέα, προφανώς η δημιουργία μιας εφαρμογής που θα μπορεί να βρίσκει κενές θέσεις παρκινγκ σε οποιοδήποτε μέρος είναι κάτι ουτοπικό, ίσως στα επόμενα χρόνια μπορεί να υλοποιηθεί. Για να εφαρμοστεί κάτι τέτοιο είναι απαραίτητη η χρήση gps του αυτοκινήτου ώστε να γνωρίζει η εφαρμογή που βρίσκεται και αν καταλαμβάνει θέση παρκινγκ. Με συνδυασμό μηχανικής μάθησης μαζί με άλλων λειτουργιών να μπορέσει να δημιουργηθεί ένα τέτοιο σύστημα. Στην Ταιβάν, υπάρχουν σε πολλά σημεία τοποθετημένες μπάρες οι οποίες ελέγχουν το χώρο γύρω τους εάν είναι κενός και μέσω μιας εφαρμογής που υπάρχει μπορεί ο χρήστης να δει που υπάρχει κενός χώρος για παρκάρισμα. Προφανώς για την υλοποίηση μιας τέτοιας ιδέας, χρειάζονται μεγάλοι οικονομικοί πόροι αλλά και εξειδικευμένο εργατικό δυναμικό.

5.5 Συμπεράσματα

Η RN μπορεί αρχικά για κάποιον να είναι δυσκατανόητη, παρόλα αυτά όταν ο χρήστης εξοικειωθεί μπορεί να δημιουργήσει περίπλοκα και όμορφα project, συν το πλεονέκτηματα ότι δεν χρειάζεται ξεχωριστός κώδικας για κάθε πλατφόρμα, με ελάχιστες γραμμές κώδικα η εφαρμογή ThessParking θα μπορούσε να τρέχει και σε iOs. Το μόνο αρνητικό που υπάρχει είναι ίσως στην εγκατάσταση και ενημέρωση κάποιων βιβλιοθηκών καθώς με μία αλλαγή που μπορεί να γίνει, υπάρχει κίνδυνος να μην τρέχει όλο το πρόγραμμα. Επίσης η χρήση του firebase σε συνδυασμό με την react λειτούργησε πολύ καλά και είναι ένα εργαλείο που δεν υστερεί σε τίποτα με άλλες βάσεις δεδομένων. Αναφορικά με την εφαρμογή, μπόρεσε να πετύχει τους στόχους που ανατέθηκαν, όπου οι σημαντικότεροι ήταν να μπορεί να χρήστης να βρει το παρκινγκ μέσω οδηγιών και η εφαρμογή του αλγόριθμου skyline operator, να προτείνει δηλαδή τα καλύτερα σε σχέση με την τιμή και την απόσταση. Ακόμα πολλές λειτουργίες που εν τέλει εφαρμόστηκαν, αρχικά φαινόταν πολύ δύσκολο να υλοποιηθούν, με σωστή χρήση βοηθημάτων και με καλύτερη κατανόηση των ιδιοτήτων κάποιων βιβλιοθηκών έγινε εφικτό.

5.6 Επίλογος

Καταληκτικά, το πρόβλημα του παρκινγκ επηρεάζεται από πολλές πτυχές, και με στοχευμένες προτάσεις μπορεί να αντιμετωπιστεί, οι λύσεις είναι πολλές το θέμα είναι να εφαρμοστούν. Το ThessParking ίσως μελλοντικά μπορέσει να βοηθήσει στην αντιμετώπιση του προβλήματος αυτού.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] Λειτουργία directions api. (Last Accessed: 7 Sep 2021)

https://developers.google.com/maps/documentation/directions/overview

[2] React native (Last Accessed: 6 Mar 2021)

https://www.stxnext.com/blog/why-use-react-native-your-mobile-app/

[3] Maciej Budziński (Last Accessed: 12 Feb 2021) What Is React Native? Complex Guide for 2021

https://www.netguru.com/glossary/react-native

[4] React Navigation (Last Accessed: 21 Feb 2021)

https://reactnavigation.org/docs/stack-navigator/

[5] React native maps (Last Accessed: 28 Aug 2021)

https://github.com/react-native-maps/react-native-maps

[6] Privacy & Policy App store (Last Accessed: 7 Jun 2021)

https://developer.apple.com/app-store/review/guidelines/#legal

[7] Privacy and Policy Play Store (Last Accessed: 16 Jul 2020)

https://support.google.com/googleplay/android-developer/answer/10144311?hl=el&ref_topic=9877467

[8] Cnc Tech (Last Accessed: 12 Jun 2020)

https://www.cnctech.gr/blog/digital-marketing/135-%CF%84%CE%B9-%CE%B5%CE%AF%CE%BD%CE%B1%CE%B9-%CF%84%CE%BF-google-analytics

[9] Adhithi Ravichandran (Last Accessed: 27 Apr 2020) Building React Native Apps — Expo or not?

https://adhithiravi.medium.com/building-react-native-apps-expo-or-not-d49770d1f5b8

[10] React-native elements (Last Accessed: 23 Aug 2021)

https://reactnativeelements.com/

[11] Firebase authentication (Last Accessed: 8 Sep 2021)

https://firebase.google.com/docs/auth

[12] Should You Include a Search Bar in Your Website Design (Last Accessed: 19 Jun 2018)

https://ny-ave.com/blog/should-you-include-a-search-bar-in-your-website-design/

[13] Visual Studio(Last Accessed: 9 Feb 2021)

https://code.visualstudio.com/docs/editor/whyvscode

[14] SHIORI YAMAZAKI (Last Accessed: 18 Aug 2020) Understanding Functional Components vs. Class Components in React

https://www.twilio.com/blog/react-choose-functional-components

[15] Constructor (Last Accessed: 1 Sep 2021)

https://reactjs.org/docs/react-component.html#constructor

[16] Stephan Borzsony, Donald Kossmann, Konrad Stocker, Univercity Munchen D-94030 Passau, Germany(Last Accessed: 1999), The Skyline Operator

https://infolab.usc.edu/csci599/Fall2007/papers/e-1.pdf

[17] Privacy and Policy (Last Accessed: 17 Jun 2019)

https://marketingplatform.google.com/about/analytics/terms/gr-20190617/

ΠΑΡΑΡΤΗΜΑ Α : Κώδικας εφαρμογής

Στο παράρτημα αυτό θα γίνει ανάλυση του κώδικα που χρησιμοποιήθηκε για την υλοποίηση της εφαρμογής. Η ανάλυση θα γίνει με χρονολογική σειρά. Δεν θα γίνει αναφορά σε όλα τα σημεία του κώδικα, αλλά στα κυριότερα που προσφέρουν σημαντικές λειτουργίες στην εφαρμογή.

Google Maps

Μετά τις παραμετροποιήσεις στις βιβλιοθήκες για την εγκατάσταση του react-native-maps, η εισαγωγή του χάρτη πραγματοποιείται με το tag MapView

```
<MapView
         followsUserLocation={true} <--------ακολουθεί τον χρήστη
         showsUserLocation={true} <----Να εμφανίζεται η τοποθεσία του χρήστη
         customMapStyle={mapstyle}
         style={styles.map} <- Οι διαστάσεις του χάρτη
         scrollEnabled={true} <-Μετακίνηση στο χάρτη
         zoomEnabled={true} <-δυνατότητα zoom in και zoom out
         pitchEnabled={true} <-Αλλαγή της γωνίας της κάμερας
         rotateEnabled={true} <-δυνατότητα περιστροφής
         initialRegion={this.state.region}
        onPress={()=>this.HideCard()}
       >
```

Κώδικας 1

Το πεδίο initialRegion είναι οι συντεταγμένες που πρέπει να μπουν ώστε να εμφανιστεί ο χάρτης στη Θεσσαλονίκη και όχι σε κάποια άλλη τοποθεσία. Τις τιμές του τις παίρνει από τον constructor που έχουν δηλωθεί οι τιμές όλων των συντεταγμένων. Η function που εκτελείται όταν ο χρήστης πατήσει πάνω στο χάρτη και όχι σε κάποιο marker, εάν οι καρτέλες των παρκινγκ είναι ενεργές τις αποκρύπτει. Child του Mapview είναι οι markers που εμφανίζουν όλες τις τοποθεσίες των παρκινγκ. Για την επεξεργασία των στοιχείων του firestore στη συγκεκριμένη περίπτωση για του markers δημιουργούνται πίνακες μέσα στο πρόγραμμά που είναι πιο εύκολα προσβάσιμοι από ότι το firestore. Παρακάτω είναι η δημιουργία του πίνακα data όπου περιέχονται όλα τα δεδομένα για τους markers. Δημιουργήθηκαν ακόμα 2 τέτοιοι πίνακες, για τους χρήστες και για την συχνότητα επίσκεψης των παρκινγκ.

```
this.subscriber=firebase.firestore().collection("data").onSnapshot(docs=>{
      let data=[]
      docs.forEach(doc=>{
        data.push(doc.data())
      })
      this.setState({data})
    })
```

Κώδικας 2

Για την προσπέλαση όλων των εγγραφών στους πίνακες χρησιμοποιείται το .map(). Οι markers περιέχουν και μία σημαντική function την markerclick διότι κάνει πολλές λειτουργίες που θα αναλυθούν στη συνέχεια

```
{this.state.defaultData.map((marker,index) => {
            return(
             <MapView.Marker onPress= {()=>this.markerClick(marker)}
              key={index}<-αναγνωριστικό για το κάθε στοιχείο
              coordinate={{ latitude: marker.Latitude,
              longitude: marker.Longitude}}<-συντεταγμένες parking
              description={marker.Description}<-Περιγραφή
              title={marker.Name}<-Όνομα parking >


                <ParkingIcon name="parking"  backgroundColor="#545BE1"
                  color="#545BE1" size={30}<-εικονίδιο parking/>

              </MapView.Marker >
            )
          }
```

Κώδικας 3

Καρτέλες των παρκινγκ

Αρχικά οι καρτέλες των παρκινγκ βρίσκονται σε tags, τα οποία τις εμφανίζουν μόνο αν είναι true. Η δημιουργία των tags έγινε με τον εξής τρόπο

```
const MyView = (props) => {
  const { children, hide, style } = props;
  if (hide) {<- κρύβει το στοιχείο εάν είναι true
    return null;
  }
  return (
    <View {...this.props} style={style}>
      { children }
    </View>
 };
```

Κώδικας 4

Παρακάτω είναι το πως γίνεται η αλλαγή των καρτελών το animation που χρησιμοποιείται, καθώς και το MyView που εξαρτάται από το state του isHidden το οποίο γίνεται false όταν ο χρήστης κάνει κλικ στο icon του παρκινγκ

```
<MyView hide= {this.state.isHidden}>
          <Animated.ScrollView<-Η κινούμενη προβολή με κύλιση που βρίσκονται οι
καρτέλες
            Horizontal<-το πως θα γίνεται το scroll
            scrollEventThrottle={1}
```

```
            showsHorizontalScrollIndicator={false}<-εμφάνιση της μπάρας κύλισης
            snapToInterval={this.state.animateCardW}<-την απόσταση που θα κάνει
scroll ανάλογα με το μέγεθος της κάρτας(αυτό αλλάζει ανάλογα εάν η κάρτα είναι σε
πλήρη προβολή)
            onScroll={Animated.event(<-το animation
              [
                {
                  nativeEvent: {
                    contentOffset: {
                      x: this.animation,

                    },
                  },
                },
              ],
              { useNativeDriver: true }
            )}
            style={styles.scrollView}<-το style που χρησιμοποιείται
            contentContainerStyle={styles.endPadding} το style που χρησιμοποιείται
για την τελευταία καρτέλα
          >
          {
            this.showNormalCard()

          }
          </Animated.ScrollView>
        </MyView>
```

Κώδικας 5

Συνάρτηση showNormalCard

Με την εκτέλεση της συνάρτησης this.showNormalCard ορίζεται αρχικά το μέγεθος των καρτελών, στην συνέχεια δημιουργούνται οι κάρτες όπου εφαρμόζεται πάλι το .map() για πρόσβαση στα δεδομένα των παρκινγκ και με index<5 ώστε να εμφανίσει μέχρι 5.

```
      <Text numberOfLines={1} style={styles.cardtitle}>
           {marker.Name}<-τίτλος parking
      </Text>

      <TouchableOpacity onPress={()=>this.startAnimation(marker)}><-    όταν    ο
χρήστης το πατάει δημιουργείται ένα animation και μεγαλώνει η καρτέλα ή μικραίνει
ανάλογα σε τι κατάσταση βρίσκεται
            <InfoIcon name='info' size={30}/>
       </TouchableOpacity>

       <TouchableOpacity>
            <Direction name="direction-sign"
            <Text> Οδηγίες</Text><- συντεταγμένες για τον προορισμό
       </TouchableOpacity >
```

Κώδικας 6

Στη συνέχεια της function ελέγχει εάν ο χρήστης είναι συνδεδεμένος ή όχι, ώστε εάν είναι να εμφανίσει το αστεράκι που μπορεί να προσθέσει το παρκινγκ στα αγαπημένα του. Μετά από τον πρώτο έλεγχο, τσεκάρει εάν αυτό το παρκινγκ υπάρχει ήδη στα αγαπημένα του χρήστη στη βάση ώστε να επιλέξει το κατάλληλο icon και να το εμφανίσει. Επίσης εάν ο χρήστης πατήσει πάνω στο αστέρι εκτελείται η συνάρτηση setFavourite η οποία ελέγχει ένα το συγκεκριμένο παρκινγκ που πάτησε ο χρήστης υπάρχει στα αγαπημένα του και πράττει αναλόγως(παρακάτω είναι η περίπτωση που υπάρχει ήδη το παρκινγκ στη βάση και πρέπει να αφαιρεθεί).

```
 for(let i=0;i<countFavourites;i++){
     if(marker.Name==index.favouriteParking[i]){<-σύγκριση εάν το όνομα του marker στη συγκεκριμένη καρτέλα είναι ίδιο με κάποιο στοιχείο από τον πίνακα favouriteParking που υπάρχει στη βάση
     userEmail.doc(doc.id)
     .update({favouriteParking:firebase.firestore.FieldValue
     .arrayRemove(marker.Name)},{merge:true})<-εντολή για αφαίρεση του parking από το firestore
     }
```

Κώδικας 7

Βαθμολογία

Εκτός από τα παραπάνω περιέχουν και άλλα στοιχεία οι καρτέλες. Το rating:

```
<View>
            <Text style={{position:'absolute',fontWeight:'bold'}}>Βαθμολογία:
            </Text>
            <Rating  fractions={1} startingValue={marker.Rating}
               onFinishRating={this.ratingCompleted} <- εκτελείται μόλις τελειώσει το rating
               readonly={this.state.canNotRate}<-ορίζεται με true false στην αρχή που προγράμματος ανάλογα αν είναι συνδεδεμένος ο χρήστης
      imageSize={15} ratingCount={5}<-Πόσα αστέρια θα έχει θα βαθμολόγηση  />
</View>
```

Κώδικας 8

Κάθε φορά που γίνεται update το πρόγραμμά εκτελείται η function που υπάρχει στην RN componentDidUpdate, έτσι εκεί γίνεται η σύγκριση εάν υπάρχει διαφοροποίηση στο συγκεκριμένο rating κι αν υπάρχει εκτελείται η setRating. Όπου ορίζει το καινούργιο rating και το αποθηκεύει στη βάση. Πρώτα γίνονται κάποιοι έλεγχοι όσον αφορά τον πίνακα στη βάση και μετά εκτελείται η async function

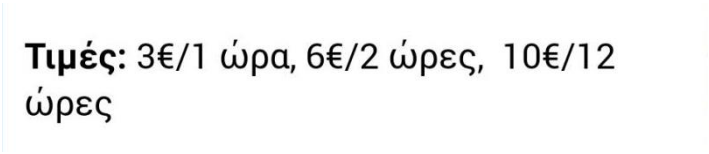
```
 async function getAll(){
```

```
              const userEmail= firebase.firestore().collection('users')<-email του
χρήστη
              const snapshot=await userEmail.where('email','==',index.email).get()
             snapshot.forEach(doc => {
              if(index.userRatings!==undefined){
                userEmail.doc(doc.id).update({userRatings:(index.userRatings)},
                       {merge:true})<-δημιουργία αν δεν υπάρχει άλλη εγγραφή
              }else{
                userEmail.doc(doc.id).set({userRatings:(userRatings)},{merge:true}
)
              }


            });
          }
          this.setState({currentRate:currentRating})<-αλλαγή του rating για την
συγκεκριμένη κατάσταση της εφαρμογής
```

Κώδικας 9

Εκτός από το rating υπάρχουν οι συναρτήσεις showPrice και showExpandCard. Η συνάρτηση showPrice ελέγχει την τιμή ανά ώρα που έχει το παρκινγκ και ανάλογα τοποθετεί 1,2 ή 3 σύμβολα του €. Η άλλη function εμφανίζει όλα τα στοιχεία που υπάρχουν διαθέσιμα για τα παρκινγκ τα οποία φαίνονται μόνο όταν ο χρήστης μεγαλώσει την καρτέλα. Τέλος έχει την συνάρτηση showAllPrices όπου παρουσιάζονται όλες οι τιμές των παρκινγκ ανάλογα με τον χρόνο που υπάρχουν(screenshot από Thessparking).



Εικόνα 0.1 Όλες οι τιμές ενός παρκινγκ όταν η κάρτα είναι μεγεθυμένη

Marker Click

Με την εκτέλεση της markerclick, η πρώτη συνάρτηση που εκτελείται είναι η saveFrequent που χρησιμοποιείται για στατιστικούς λόγους. Η μεταβλητή today δημιουργείται στην αρχή της εφαρμογής, με την χρήση της βιβλιοθήκης moment, παίρνουμε την σημερινή ημερομηνία.

```
 async function saveFrequent(){
const parkingMarker= firebase.firestore().collection('data')
    const snapshot=await parkingMarker.where('Name','==',marker.Name).get()<-
παίρνει τα parking είναι ίδια με το clicked Parking
    snapshot.forEach(doc => {
      const data = doc.data();
      const temp=firebase.firestore().collection('frequent').doc(doc.id)
```

```
        temp.get().then(doc=>{
          if(doc.exists){
            let checkValues=false<-αν γίνει true σημαίνει υπάρχει ήδη εγγραφή
            let value=doc.data()
            let counter=doc.data()
            let temp2=doc.data()['date']  <-ημερομηνία και τιμή
            value=value['date'][0]['value']<-ημερομηνία
            counter=counter['date'][1]['counter']<-τιμή

           for(let i=0;i<value.length;i++){<-ελέγχει αν υπάρχει ήδη το  παρκινγκ
την συγκεκριμένη ημερομηνία στη βάση για να αυξήσει τον counter
             if(value[i]==today){
               let setCounter=counter[i]+1
               temp2[1].counter[i]=setCounter
               checkValues=true
             }
           }
           if(checkValues==false){<-εκτελείται αν δεν υπάρχει εγγραφή με αυτά τα
στοιχεία αλλά υπάρχει πίνακας date
            temp2[0].value[value.length]=today
            temp2[1].counter[value.length]=1
          }
           firebase.firestore().collection('frequent').doc(doc.id)
            .set({date:temp2},{merge:true})
          }else{<-εκτελείται αν δεν υπάρχει τίποτα προηγουμένως
            firebase.firestore().collection('frequent')
             .doc(doc.id).set({date:[{value:[today]},{counter:[1]}]})
          }
        })

  })}
```

Κώδικας 10

Η επόμενη λειτουργία εκτελείται όταν ο χρήστης κάνει scroll στις καρτέλες και πατήσει ένα άλλο parking, αυτό αυτομάτως να πηγαίνει στη πρώτη θέση των καρτελών. Επίσης εκτελείται και η παρακάτω συνάρτηση που ουσιαστικά όταν αλλάζει η καρτέλα με το parking αλλάζει και το επιλεγμένο παρκινγκ δηλαδή ο τίτλος που φαίνεται. Επιπρόσθετα και όταν κάνει κλικ σε άλλο παρκινγκ να εμφανίζεται πάλι ο τίτλος.

```
 onRegionChangeComplete = () => {//used to show the map title with the use of Ref

    if (this.myRef && this.myRef.current && this.myRef.current.showCallout ) {
      this.myRef.current.showCallout();

    }
  };
var temp = this.state.data.filter(x=> x.Name != marker.Name);
```

```
  temp.unshift(marker);
```
Κώδικας 11

Τέλος αλλάζει την κατάσταση της τιμής isHidden ώστε να εμφανιστεί η κάρτα, και επίσης αλλάζει την σειρά στον πίνακα δεδομένων έτσι ώστε η πρώτη τιμή του πίνακα να είναι το συγκεκριμένο παρκινγκ που έγινε κλικ.

Μπάρα αναζήτησης

Πριν από την δημιουργία του search Bar, φτιάχτηκαν δύο μεταβλητές searchBarFocused που γίνεται true εάν το search Bar είναι ενεργοποιημένο, και η keyboardStatus που αναφέρεται στο πληκτρολόγιο.

```
  <TextInput
            placeholder="Βρες Parking" <-Ο τίτλος που υπάρχει
            onChangeText={text=>{this.searchParking(text)}}
             key='SearchBar'
            placeholderTextColor="#000"
            autoCapitalize="none"<-μετατροπή σε κεφαλαία
            textAlign="left"
            onSubmitEditing={Keyboard.dismiss}<-όταν πατηθεί enter να φύγει το
πληκτρολόγιο
            style={{flex:1,padding:0}}/>
```
Κώδικας 12

Με την εκτέλεση της function searchParking γίνεται true το searchBarFocused, και σε ένα πίνακα που έχει δημιουργηθεί πιο πριν (parkingFilter) περνάει σαν τιμή αυτό που υπάρχει στο searchBar και εκτελείται παράλληλα και ο παρακάτω κώδικας. Όπου παίρνει το στοιχείο που υπάρχει στον πίνακα και το συγκρίνει με τα ονόματα παρκινγκ που υπάρχουν. Ακόμα εάν πατηθεί κάποιο στοιχείο της λίστας εκτελείται η handlerSimpleCall που απενεργοποιεί το πληκτρολόγιο, εκτελεί την function markerclick για να εμφανιστεί η καρτέλα του παρκινγκ που πατήθηκε και αφήνει κενό τον πίνακα parkingFilter.

```
 {this.state.parkingFilter.map((marker,index)=>(

          <ListItem
          onPress={() =>{this.handlerSimpleCall(marker)}}
           key={index}   >
          <ListItem.Content>
            <ListItem.Title style={{position:'absolute',alignContent:'center',
                alignItems:'center',alignSelf:'center'}}>{marker.Name}
            </ListItem.Title>

          </ListItem.Content>
        </ListItem>
            ))}
```
Κώδικας 13

Navigation

Το navigation πραγματοποιείται στο App.js όπου γίνεται η εναλλαγή των οθονών από την μία στην άλλη. Στην αρχή ελέγχεται εάν υπάρχει συνδεδεμένος χρήστης και μετά γίνονται οι έλεγχοι για να εμφανιστεί ο κατάλληλος drawer, δηλαδή το navigation που μπορεί να κάνει ο χρήστης με το να τραβήξει την οθόνη προς τα δεξιά.

```
if(!loaded){<-μέχρι να φορτωθεί η εφαρμογή εμφανίζεται ένας loader

      return(
          <View style={[styles.container, styles.horizontal]} >
              <ActivityIndicator size="large" color='#545BE1' />
            </View>
              )
  } if(!loggedIn){<-εάν δεν είναι συνδεδεμένος πηγαίνει σε αυτόν το drawer
      return (
        <NavigationContainer>
          <RegisterDrawer/>
        </NavigationContainer>
     )
    }if(this.state.Owner==true){<-εάν είναι ιδιοκτήτης πηγαίνει στον drawer για
τους ιδιοκτήτες
      return(
        <NavigationContainer>
          <LoginOwnerDrawer/>
        </NavigationContainer>
    )
    }if(this.state.Owner==false){
      return(
        <NavigationContainer>
          <LoginDrawer/>
        </NavigationContainer>
      )
    }
```

Κώδικας 14

Παρακάτω παρουσιάζεται το navigation για συνδεδεμένο χρήστη που είναι ιδιοκτήτης

```
function LoginOwnerDrawer(){//when user login and is Owner
  return(
      <Drawer.Navigator drawerContent={props => <CustomDrawerContent {...props} />
} ><-Εδώ δημιουργείται το κουμπί έξοδος που όταν πατηθεί ο χρήστης αποσυνδέεται
        <Drawer.Screen name='Ιδιοκτήτης' component={Navigator}options={{drawerIcon
: ({ tintColor }) =>
        (<AntDesign name="linechart" color="#545BE1" size={25}/>),}}/><-η οθόνη με
τα γραφήματα και τις διαφημίσεις για τους ιδιοκτήτες
        <Drawer.Screen name="GoogleMaps" component={GoogleMapScreen}
        options={{drawerIcon: ({ tintColor }) =>
        (<Foundation name="map" color="#545BE1" size={25}/>),}}/>
```

```
          <Drawer.Screen name="Είσοδος Χρήστη" component={LoginScreen} <-οθόνη    με
τους χάρτες
           options={{drawerIcon: ({ tintColor }) =>
           (<Ionicons name="ios-enter" color="#545BE1"size={25}/>),}}/>

          {(props) =><Drawer.Screen name="Δημιουργία Χρήστη"
            component={RegisterScreen} {...props}
          options={{drawerIcon: ({ tintColor }) =>
          (<AntDesign name="adduser" color="#545BE1" size={25}/>),}}/>}<-οθόνη   για
την δημιουργία χρήστη

        </Drawer.Navigator>

  )

}
```

Κώδικας 15

Επιπρόσθετα υπάρχουν οι function setOnlyUser και setTypeUser όπου εκτελούνται κάθε φορά που γίνεται rerender, και ελέγχουν εάν το χρήστης είναι ιδιοκτήτης parking ή όχι (παρακάτω είναι η setTypeUser).

```
firebase.firestore().collection("users")
.get().then(docs=>{//get collection of user data and passes to users array
    let users=[]
    docs.forEach(doc=>{
      users.push(doc.data())
     const user = firebase.auth().currentUser;//to get current user that is logged

     if(user!==null){
      if(doc.data()['email']==user.email&&doc.data()['isOwner']==true){
        this.setState({Owner:true})
      }
```

Κώδικας 16

Είσοδος και εγγραφή

Η οθόνη για την σύνδεση του χρήστη και η οθόνη για την δημιουργία του χρήστη, είναι παρόμοιες απλά η μία δημιουργεί τον χρήστη και η άλλη τον συνδέει στην εφαρμογή. Αρχικά για το log in screen εκτός από τα πεδία που περιέχει για να συνδεθεί ο χρήστης στην εφαρμογή έχει :

```
<Button onPress={() => this.onSignUp()}<-εκτέλεση της συνάρτησης που συνδέει τον
χρήστη
 title="Εισοδος "
 color='#545BE1'
/>
 <TouchableOpacity onPress={()=>this.moveToForgot()} style={{paddingTop:'4%'}}>
      <Text style={{fontSize:15,color:'#F98A29'}}>Ξεχάσατε τον κωδικό σας;
```

```
      </Text>
</TouchableOpacity><-μεταφορά στην οθόνη όπου ο χρήστης μπορεί να αλλάξει κωδικό
<TouchableOpacity onPress={()=>this.moveToRegister()} style={{paddingTop:'4%'}}>
      <Text style={{fontSize:15,color:'#0E9FFF'}}>Δεν έχετε λογαριασμό;Πατήστε εδ
ώ για την δημιουργία λογαριασμού
      </Text>
</TouchableOpacity><-μεταφορά στην οθόνη register
```

Παρακάτω φαίνονται κάποιοι από τους ελέγχους που κάνει η συνάρτηση onSignUp:

```
if(!this._isValidEmail(email)){<-έλεγχος αν το email είναι έγκυρο με την συνάρτηση
isValid
            this.setState({setError:"Παρακαλώ βάλτε έγκυρο E-mail"})
            return
        }
        firebase.auth().signInWithEmailAndPassword(email, password)
            .then((result) => {<-εκτέλεση εάν τα στοιχεία ήταν σωστά
                this.setState({setError:null})
                alert('Welcome')
            })
            .catch((error) => {<-error εάν δεν υπάρχει το email
                if (error.code === 'auth/user-not-found') {
                    this.setState({setError:"Το συγκεκριμένο E-
mail δεν υπάρχει. Παρακαλώ δοκιμάστε με άλλο"})
                  }
```

Κώδικας 17

Το register δουλεύει με παρόμοιο τρόπο, απλά χρησιμοποιεί λίγες function παραπάνω και στην function onSignUp περιέχει περισσότερους ελέγχους.

```
PassInputChange(password){<-χρησιμοποιείται ώστε αν πληροί τα κριτήρια για σωστό
κωδικό πρόσβασης να εμφανίζεται ένα tick δίπλα από τον κωδικό
        if(password.length!==0){
            this.setState({ password:password })
            if(password.length>5){
                this.setState({check_passInputChange:true})
            }else{
                this.setState({check_passInputChange:false})
            }
        }
    }
    checkPass(pass){<-συγκρίνει τα 2 πεδία αν οι κωδικοί είναι ίδιοι
        if(pass==this.state.password){
            this.setState({confirm_pass:true})
            this.setState({confirm_password:pass})
        }else{
            this.setState({confirm_pass:false})
            this.setState({confirm_password:pass})
        }
    }
```

```
 <TextInput
                        placeholder="Κωδικός"
                        secureTextEntry={true}<-ώστε να μην φαίνεται ο κωδικός
                        onChangeText={(password) =>{ this.PassInputChange(password
),this.state.setError}}<-όταν αλλάζει τιμή να εκτελείται
                        style={styles.textInput}
                        error={this.state.isValid}<-εάν είναι false ενεργοποιείται
το error
                    />
```

Κώδικας 18

Στατιστικά και διαφημίσεις

Γραφήματα

Στην οθόνη που παρουσιάζονται τα στατιστικά υπάρχουν πολλές function και λειτουργίες. Εδώ να σημειωθεί πως είναι το μόνο αρχείο που χρησιμοποιήθηκε function component από ότι class component. Για την εναλλαγή των ημερομηνιών χρησιμοποιούνται οι συναρτήσεις getSunday(), getSaturday όπου σε συνδυασμό δίνουν τις ημέρες της εβδομάδας, και η changeData όπου ενεργοποιείται όταν πατηθεί το δεξί βελάκι που προσθέτει 7 ημέρες. Αντίστοιχη function υπάρχει για το αριστερό βελάκι το μειώνει τις μέρες.

Αποτέλεσμα συνάρτησης moment

```
 function changeData() {
        setSunday(moment(Sunday).add(7, 'days').toDate());
        setSaturday(moment(Saturday).add(7, 'days').toDate());
      }
```

Κώδικας 19

Ακολουθεί μέρος στο πως δημιουργούνται τα γραφήματα.

```
 <YAxis<-άξονας Y
   data={props.data}<-δεδομένα του γραφήματος
   min={props.ymin}<-ελάχιστο άξονα Y
   max={props.ymax}<-μεγιστο άξονα Y
   contentInset={verticalContentInsetYLabel}
   svg={axesSvg}<-style της γραμμής στον άξονα Y
   numberOfTicks={props.ticks}<-πόσες τιμές θα υπάρχουν στον άξονα Y
   formatLabel={(value,index)=>value}>
</YAxis>
<BarChart<-γράφημα με μπάρες
   yMax={props.ymax}<-ελάχιστο γράφημα
   yMin={props.ymin}<-μεγιστο γράφημα
   style={{ flex: 1 }}
```

```
   data={props.data}<-δεδομένα
   svg={{ fill: '#FFFFFF' }}<-γραμμές
   contentInset={verticalContentInset}
   spacingInner={0.3} >
```

Κώδικας 20

Παρακάτω παρουσιάζεται το πως εμφανίζονται τα στατιστικά καθώς και πως υπολογίζεται ο μέσος όρος. Η showStatistics εκτελείται μόνο όταν το toggleSwitch ενεργοποιείται, όπως επίσης με toggleSwitch λειτουργεί και η εναλλαγή των γραφημάτων

```
function showStatistics(){//show Stats
        if(isEnabled2==false){
            setHidden(false)
        }
        else{
            setHidden(true)
        }
    }

         //Average
         var total = 0;
         for(var i = 0; i < props.data.length; i++) {
             total += props.data[i];
         }
         var avg = total / props.data.length;
        avg= parseFloat(avg).toFixed(0);
```

Κώδικας 21

Ακόμα για να εμφανιστούν πόσοι χρήστες έχουν βάλει το παρκινγκ του χρήστη στα αγαπημένα τους εκτελείται η checkFavouriteUsers.

```
function checkFavouriteUsers(){//used to see the owner how many users had set his
parking to their favourites

      user = firebase.auth().currentUser;
      async function saveFavourite(){<-δημιουργία async function για πρόσβαση στα
δεδομένα του firestore
         const snapshot=await firebase.firestore().collection('users')
          .where('email' ,'==',user.email).get()<-χρήστης που είναι συνδεδεμένος
            snapshot.forEach(doc => {
              setParkingOwned(doc.data()['parkingOwned'])<-ποιο parking έχει
              firebase.firestore().collection('users').get().then((querySnapshot)=
>{  <-αναζήτηση σε όλους τους χρήστες
                const objectsArray = [];
                querySnapshot.forEach((user) => {
                  objectsArray.push(user.data());
                });
                objectsArray.map((index,key)=>{
                  if(index.favouriteParking!==undefined){<-εάν υπάρχει το πεδίο
```

```
                    for(let i=0;i<index.favouriteParking.length;i++){
                      if(index.favouriteParking[i]==doc.data()['parkingOwned']){<-
αναζήτηση αν υπάρχει το parking και έχει ήδη πατήσει μια φορά απλα αυξάνεται ο
counter
                        counter=counter+1
                        setCounterFavourites(counter)
                      }
```

Κώδικας 22

Για να αλλάξουν τα δεδομένα στο γράφημα όταν ο χρήστης κάνει κλικ

```
function showUserStatistics(){//show data function
        user = firebase.auth().currentUser;
        data=[]
        let id=null<- το id που θα χρησιμοποιηθεί για να βρεθεί το parking
        let parking=null
        async function findParking(){//find which parking user that is logged owns
          const snapshot1=await firebase.firestore().collection('users')
            .where('email' ,'==',user.email).get()
            snapshot1.forEach(doc => {
              parking=doc.data()['parkingOwned']
            })

          const snapshot=await firebase.firestore().collection('data')
            .where('Name' ,'==',parking).get()//get parking id
          snapshot.forEach(doc => {
            id=doc.id

          });
         const dataGraph= await firebase.firestore()
          .collection('frequent').doc(id).get()
         const dates=dataGraph.data()['date'][0]['value']//dates that parking icon
 had been pressed
        const counters=dataGraph.data()['date'][1]['counter']//times that parking
 icon had been pressed in each date
        if(data.length<7){//set values in data table
           for(let i=0;i<getArrayDate.length;i++){
               for(let j=0;j<dates.length;j++){
                if(dates[j]==getArrayDate[i]){//if exists set value
                  data.push(counters[j])
                  break;
                }else{
                  if(j==dates.length-1){//if not exists set value 0
                    data.push(0)
```

```
                }

              }
            }
          }
```

Κώδικας 23

Διαφημίσεις

Οι διαφημίσεις στην εφαρμογή μπορούν να προβληθούν με 2 τρόπους, αρχικά ακριβώς από κάτω φαίνεται το BannerAd

```
<BannerAd

          unitId={TestIds.BANNER}<-το banner που θα προβληθεί, το συγκεκριμένο
είναι το default
          size={BannerAdSize.SMART_BANNER}<-το μέγεθος του banner
          requestOptions={{
            requestNonPersonalizedAdsOnly: true<-Να μην είναι προσωποποιημένες οι
διαφημίσεις
          }}
          onAdLoaded={() => console.log('Advert loaded')}<-όταν φορτώσει η
διαφήμιση
          onAdFailedToLoad={(error) => {<-Αν αποτύχει να φορτώσει
            console.error('Advert failed to load: ', error)
          }}
        />
```

Κώδικας 24

Για τους άλλους 2 τύπους διαφημίσεων καλούνται functions για να εμφανιστούν:

```
 showInterstitialAd = () => {
          // Create a new instance
          const interstitialAd = InterstitialAd.createForAdRequest(TestIds.INTERST
ITIAL);

          // Add event handlers
          interstitialAd.onAdEvent((type, error) => {
            if (type === AdEventType.LOADED) {
              interstitialAd.show();
            }
          });}
```

Κώδικας 25

Δημιουργία συνάρτησης βελτιστοποίησης

Ο αλγόριθμος βελτιστοποίησης εκτελείται όταν ενεργοποιηθεί το switch. Ο πρώτος έλεγχος που εκτελείται φαίνεται παρακάτω, όπου παίρνει τις συντεταγμένες του χρήστη.

```
let maxDistance=0

    const checkUser = async () => {

     let location= await RNLocation.getLatestLocation({timeout: 100})//user locati
on
      if(location!==null){
        userLocationLat=location.latitude
        userLocationLon=location.longitude
      }



    }
    {checkUser()}
    if(userLocationLat==null){
      Alert.alert('','Παρακαλώ ενεργοποιήστε τις πληροφορίες τοποθεσίας')
      return(null)
    }
```

Κώδικας 26

Εδώ φαίνεται πως κατηγοριοποιούνται τα πορτοκαλί παρκινγκ. Με το getArrayMin παίρνει την μικρότερη τιμή του πίνακα και την συγκρίνει με όλες τις εγγραφές. Παρουσιάζεται τόσο η σύγκριση προς την απόσταση όπου αν είναι ίδια πάει και συγκρίνει με την τιμή.

```
for(let i=0;i<this.state.bestParkings.length;i++){//categorize orange parkings
      let temp=false
      let minDistance=getArrayMin(this.state.orangeDistance)<-ελάχιστο πίνακα
      if(this.state.orangeParkings.length<5){<-εισαγωγή 5 πρώτων μεταβλητών
        this.state.orangeParkings.push(this.state.bestParkings[i])
        this.state.orangeDistance.push(this.state.parkingDistance[i])
        this.state.orangePrice.push(this.state.parkingPrice[i])
      }else if(this.state.bestParkings.length>=5){<-έλεγχος για να μείνουν τα 5 με
την μεγαλύτερη απόσταση
        for(j=0;j<this.state.orangeParkings.length;j++){<-έλεγχος για να μην
υπάρχουν διπλότυπα στον πίνακα
          if(this.state.orangeParkings[j]==this.state.bestParkings[i]){
            temp=false
            break;
          }else{
            temp=true

          }
        }
        if(this.state.parkingDistance[i]>minDistance){
          for(let j=0;j<this.state.orangeParkings.length;j++){
```

```
            if(this.state.orangeDistance[j]==minDistance&&temp==true){
              this.state.orangeParkings[j]=this.state.bestParkings[i]
              this.state.orangeDistance[j]=this.state.parkingDistance[i]
              this.state.orangePrice[j]=this.state.parkingPrice[i]
              break;
            }
          }
        }else if(this.state.parkingDistance[i]==minDistance){<-αν είναι ίσα
έλεγχος με βάση την τιμή
          for(let j=0;j<this.state.orangeParkings.length;j++){

            if((this.state.orangeDistance[j]==minDistance&&temp==true )){
              let tempParking=this.state.orangeParkings[j]
              let tempDistance=this.state.orangeDistance[j]
              let tempPrice=this.state.orangePrice[j]
              for(let z=0;z<this.state.orangeParkings.length;z++){
                if(tempDistance==this.state.orangeDistance[z]
                      &&tempParking!==this.state.orangeParkings[z]
                      &&tempPrice<=this.state.orangePrice[z]){
                  this.state.orangeParkings[z]=this.state.bestParkings[i]
              this.state.orangeDistance[z]=this.state.parkingDistance[i]
              this.state.orangePrice[z]=this.state.parkingPrice[i]
              break
                }
              } if(this.state.orangePrice[j]<=this.state.parkingPrice[i]){
                this.state.orangeParkings[j]=this.state.bestParkings[i]
              this.state.orangeDistance[j]=this.state.parkingDistance[i]
              this.state.orangePrice[j]=this.state.parkingPrice[i]

              break;
              }
              break
            }
          }
        }
      }
    }
```

Κώδικας 27

Για να αλλάξουν χρώμα τα εικονίδια των παρκινγκ εφαρμόζεται ο εξής τρόπος. Και έτσι γίνεται και με τα υπόλοιπα. Επίσης μόλις απενεργοποιείται το switch ξαναγίνονται όλα μπλε

```
 for(let i=0;i<this.state.orangeParkings.length;i++){//check orange parkings when
switch is enable
            if(this.state.orangeParkings[i]==marker.Name){
              results.push(
                <MapView.Marker onPress= {()=>this.markerClick(marker)}

                    key={index}
```

```
                    coordinate={{ latitude: marker.Latitude,
                    longitude: marker.Longitude}}
                    description={marker.Description}
                    title={marker.Name} >


                  <ParkingIcon name="parking"   color='#F7A10B'  size={30}/>

                </MapView.Marker >);
              exists=true
              break

            }
          }
```

Κώδικας 28

Directions

Η function που δίνει οδηγίες εκτελείται μόλις ο χρήστης κάνει κλικ στο κουμπί οδηγίες και έχει ενεργή την τοποθεσία του. Επίσης για το κουμπί που σταματάει τις οδηγίες, υπάρχει συνάρτηση που θέτει το origin και destination με null

```
showdirections(marker,location){
    this.setState({showWay:false})<-όταν false δείχνει την διαδρομή
    this.setState({origin:{latitude:userLocationLat,longitude:userLocationLon}})<-
user
    this.setState({destination: {latitude:marker.Latitude,longitude:marker.Longitu
de}})<-parking
    this.setState({region:{latitudeDelta:0.0122,latitude:userLocationLat,longitude
:userLocationLon,longitudeDelta:0.0122*ASPECT_RATIO}})<-αλλαγή region για να κάνει
focus στον χρήστη
    let temp1=this.state.origin
    let temp2=this.state.destination
   this.map.animateToRegion({latitude:this.state.region.latitude,longitude:this.st
ate.region.longitude,
    latitudeDelta:this.state.region.latitudeDelta,longitudeDelta:this.state.region
.longitudeDelta})<-να ακολουθεί τον χρήστη
    if(temp1==temp2){<-αν είναι ίσα έφτασε ο χρήστης τον προορισμό του
      Alert.alert('','Φτάσατε στον προορισμό σας!')
      this.setState({origin:null})
      this.setState({destination:null})

    }
  }
```

Κώδικας 29

Τέλος η εμφάνιση της διαδρομής γίνεται με τον εξής τρόπο

```
 <MapViewDirections
          origin={this.state.origin}
          destination={this.state.destination}
          apikey={GOOGLE_MAPS_APIKEY} // insert your API Key here
          mode='DRIVING'
          strokeWidth={5}
          strokeColor="#545BE1"
```

Κώδικας 30

Δημιουργία καρτέλας αγαπημένα:

```
findFavourite(){//find favourite parkings of the user
        let user = firebase.auth().currentUser;//to get current user that is logge
d in
        this.state.users.map((index,key)=>{
            if(user!==null){
                const email=user.email<-αν είναι συνδεδεμένος ο χρήστης
                let counter=0<-για τις επαναλήψεις
                if(index.email==user.email){
                    for(let i=0;i<index.favouriteParking.length;i++){
                        if(counter!==index.favouriteParking.length){
                            favouritePark.push(index.favouriteParking[i])
                            counter=counter+1
                        }
                        if(favouritePark.length==index.favouriteParking.length){//
if length of the array is equal with the length of the array that exists if the da
tabase show favourite
                            this.setState({loaded:true})
                            break;
                        }

                    }
                }
            }

        })
    }
```

Κώδικας 31

Φίλτρα τοποθεσίας και τιμών

Αρχικά όλα τα στοιχεία και των δύο φίλτρων(διακόπτες και τίτλος μπήκαν) μέσα στο Accordian το οποίο είναι ένα component που φτιάχτηκε.

```
return (
       <View key={Math.floor(Math.random() * 100) + 1 }>
            <TouchableOpacity ref={this.accordian}
             style={styles.row} onPress={()=>this.toggleExpand()}><-συνάρτηση που
μεγαλώνει και μικραίνει το touchableopacity
               <Text style={[styles.title, styles.font]}>{this.props.title}</Text
>
               <Icon name={this.state.expanded ? 'keyboard-arrow-up' : 'keyboard-
arrow-down'} size={30} style={{paddingRight:'10%'}} />
            </TouchableOpacity><-αλλαγή εικονιδίου
            <View style={styles.parentHr}/>
            {
                this.state.expanded &&
                <View style={styles.child}>
                    <Text>{this.props.data}</Text>
                </View>
            }
       </View>
    )
  }
  toggleExpand=()=>{
    LayoutAnimation.configureNext(LayoutAnimation.Presets.easeInEaseOut);
    this.setState({expanded : !this.state.expanded})
  }

}
```

Κώδικας 32

Για να δημιουργηθούν ένα φίλτρο αρχικά δηλώνεται έτσι

```
<Accordian title={'Τιμές'} key={2} data={filtersPrices}>
</Accordian>
```

Κώδικας 33

Και έπειτα εκεί που είναι filtersPrice τοποθείται ένα list item μαζί με ένα text και ένα switch και κάθε φορά που ο χρήστης πατάει το switch εκτελείται η εκάστοτε συνάρτηση.

```
showOnlyLowPrice(){
    if(this.state.enableLowPrice==false){<-false γιατί ακόμα δεν έχει
ενεργοποιηθεί ο διακόπτης
      if(this.state.enableKentro||this.state.enableToumba||this.state.enableKentro
){<-εάν καποιος από τους διακόπτες των τοποθεσιών είναι ενεργός απενεργοποιείται
        this.setState({enableKentro:false})
        this.setState({enableXarilaou:false})
```

```
        this.setState({enableToumba:false})
      }
      if(this.state.enableHighPrice==true){
        this.setState({enableHighPrice:false})

      }<-εάν ο άλλος διακόπτης το τιμών είναι ενεργός απενεργοποιείται
        const array=this.state.arrayRegion.filter(obj=>obj.Price<=2)
        this.setState({data:array})<-το arrayRegion ισούται με το data των parking απλά χρησημοποιείται γιατί τα δεδομένα του δεν αλλάζουν. Επίσης με την συνάρτηση filter επιλέγονται μόνο οι εγγραφές που πληρούν τα κριτήρια. Στη συγκεκριμένη περίπτωση μικρότερο ή ίσο με το 2

    }else{
      this.setState({data:this.state.arrayRegion})<-άμα απενεργοποιηθεί ο διακόπτης, τα δεδομένα επιστρέφουν στη αρχική τους μορφή
    }
  }
```

Κώδικας 34

Κρατήσεις

Για τις κρατήσεις δημιουργήθηκε ξεχωριστό component που απλά έγινε η εισαγωγή του στις καρτέλες των χαρτών και έπρεπε να περαστούν κάποιες τιμές για να λειτουργήσει(ημερομηνία, πλάτος, όνομα παρκινγκ, χρήστης, διεύθυνση).

```
onChange = (event, selectedDate) => {<-η συνάρτηση αυτή εκτελείται όταν εμφανίζεται το ημερολόγιο και ο χρήστης πατήσει πάνω σε κάποια άλλη ημερομηνία από την επιλεγμένη

    const currentDate = selectedDate || this.state.date;<-επιλέγεται η ημερομηνία που επίλεξε ο χρήστης
    this.setState({date:currentDate})

    if(event.type=='set'){<-εάν ο χρήστης πατήσει οκ τότε εμφανίζεται η ώρα για να επιλέξει ο χρήστης
        this.setState({showTime:true})
    }
    this.setState({show:false})
  };
  onChangeTime=(event, selectedDate)=> { η συνάρτηση αυτή εκτελείται όταν εμφανίζεται η ώρα και ο χρήστης πατήσει πάνω σε κάποια άλλη ημερομηνία από την επιλεγμένη
```

```
   const currentDate = selectedDate || this.state.date;

   if(event.type=='set'){<-εάν πατήσει οκ
    this.setState({date:currentDate})
    const reservation=moment(this.state.date).format('D M YYYYHH:mm')<-αλλάζει
μορφή η ημερομηνία με την χρήση της moment
    const date=moment(this.state.date).format('D M YYYY')
    const hour=moment(this.state.date).format('HH:mm')
    const address=this.props.address<-η τιμή της διεύθυνσης παίρνετε από το
επιλεγμένο παρκινγκ
    firebase.firestore().collection('reservations')
       .add({date:date,hour:hour,email:this.props.user.email,
       parking:this.props.markerName})<-αποθήκευση των δεδομένων στον πίνακα
reservations
    .catch((error)=>{
      Alert.alert('','Παρουσιάστηκε σφάλμα παρακαλώ προσπαθήστε αργότερα')

    })
    Alert.alert('','Η κράτηση πραγματοποιήθηκε επιτυχώς για την ημερομηνία'+' '+da
te+' '+'και ώρα'+' '+hour+"\n"+'Η οδός του Parking είναι'+' '+address)<-
εμφανίζεται εάν είναι επιτυχής η κράτηση
   }
```

Κώδικας 35