Πανεπιστήμιο Δυτικής Αττικής
Τμήμα Μηχανικών Πληροφορικής και Υπολογιστών

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Σχεδιασμός και ανάπτυξη συστήματος διαχείρισης επισκευών αυτοκίνητων

Φοιτητής: Κωνσταντίνος Σταθακόπουλος (Α.Μ. 711161041)
Επιβλέποντες καθηγητές: κ. Βογιατζής Ιωάννης, κ. Μελετίου Γεώργιος

Αθήνα 2022

Copyright © Με επιφύλαξη παντός δικαιώματος. All rights reserved.

ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ και Σταθακόπουλος Κωνσταντίνος, Ιούνιος, 2022

Απαγορεύεται η αντιγραφή, αποθήκευση και διανομή της παρούσας εργασίας, εξ ολοκλήρου ή τμήματος αυτής, για εμπορικό σκοπό. Επιτρέπεται η ανατύπωση, αποθήκευση και διανομή για σκοπό μη κερδοσκοπικό, εκπαιδευτικής ή ερευνητικής φύσης, υπό την προϋπόθεση να αναφέρεται η πηγή προέλευσης και να διατηρείται το παρόν μήνυμα. Ερωτήματα που αφορούν τη χρήση της εργασίας για κερδοσκοπικό σκοπό πρέπει να απευθύνονται προς τους συγγραφείς.

Οι απόψεις και τα συμπεράσματα που περιέχονται σε αυτό το έγγραφο εκφράζουν τον/την συγγραφέα του και δεν πρέπει να ερμηνευθεί ότι αντιπροσωπεύουν τις θέσεις του επιβλέποντος, της επιτροπής εξέτασης ή τις επίσημες θέσεις του Τμήματος και του Ιδρύματος.

Kleio Sgouropoulou
Digitally signed by Kleio Sgouropoulou
Date: 2023.01.09 11:58:48 +02'00'

Ioannis Voyiatzis
Digitally signed by Ioannis Voyiatzis
Date: 2022.11.08 11:49:11 +02'00'

Georgios Meletiou
Digitally signed by Georgios Meletiou
Date: 2022.11.07 11:16:00 +02'00'

ΔΗΛΩΣΗ ΣΥΓΓΡΑΦΕΑ ΠΤΥΧΙΑΚΗΣ/ΔΙΠΛΩΜΑΤΙΚΗΣ ΕΡΓΑΣΙΑΣ

Ο κάτωθι υπογεγραμμένος Σταθακόπουλος Κωνσταντίνος του Δημητρίου, με αριθμό μητρώου 711161041 φοιτητής του Πανεπιστημίου Δυτικής Αττικής της Σχολής ΜΗΧΑΝΙΚΩΝ του Τμήματος ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ,

δηλώνω υπεύθυνα ότι:

«Είμαι συγγραφέας αυτής της πτυχιακής/διπλωματικής εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, οι όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών ή λέξεων, είτε ακριβώς είτε παραφρασμένες, αναφέρονται στο σύνολό τους, με πλήρη αναφορά στους συγγραφείς, τον εκδοτικό οίκο ή το περιοδικό, συμπεριλαμβανομένων και των πηγών που ενδεχομένως χρησιμοποιήθηκαν από το διαδίκτυο. Επίσης, βεβαιώνω ότι αυτή η εργασία έχει συγγραφεί από μένα αποκλειστικά και αποτελεί προϊόν πνευματικής ιδιοκτησίας τόσο δικής μου, όσο και του Ιδρύματος.
Παράβαση της ανωτέρω ακαδημαϊκής μου ευθύνης αποτελεί ουσιώδη λόγο για την ανάκληση του πτυχίου μου».

Ο Δηλών
Σταθακόπουλος Κωνσταντίνος

Πρόλογος

Με την εξέλιξη της τεχνολογίας, τα κινητά τηλέφωνα έχουν αποκτήσει περισσότερες δυνατότητες διευρύνοντας με αυτό τον τρόπο τις υπηρεσίες που μπορούν να παρέχουν εκτός από τις βασικές λειτουργίες των τηλεφώνων (επικοινωνία). Αυτό συμβαίνει διότι αναπτύσσονται εφαρμογές από τους προγραμματιστές οι οποίες καθιστούν τις συσκευές τηλεφωνίας «έξυπνα» τηλέφωνα (smartphones). Επομένως, εκτός από την επικοινωνία, με τις επιπλέον εφαρμογές διευκολύνουν την καθημερινότητα των χρηστών που τις χρησιμοποιούν, εκτελώντας πολλές χρήσιμες λειτουργίες σε μια συσκευή με μεγάλη φορητότητα. Τα έξυπνα κινητά τηλέφωνα μπορούν να τρέχουν διαφορετικά λειτουργικά συστήματα όπως το Android της Google και το iOS της Apple αλλά υπάρχουν και άλλα λιγότερο γνωστά στο ευρύ κοινό. Αυτή η διπλωματική εργασία εστιάζει στην ανάπτυξη μιας εφαρμογής για Android η οποία διαχειρίζεται τα δεδομένα που διατηρούνται σε ένα βιβλίο επισκευών για αυτοκίνητα και παρέχει πρόσθετες δυνατότητες για να εξυπηρετεί τον ιδιοκτήτη ενός ή περισσότερων αυτοκινήτων.

Περίληψη

Η παρούσα διπλωματική εργασία έχει στόχο την σχεδίαση και την ανάπτυξη συστήματος διαχείρισης επισκευών αυτοκινήτων (ψηφιακό βιβλίο επισκευών αυτοκινήτων) αναπτύσσοντας μια εφαρμογή για έξυπνα κινητά τηλέφωνα. Η ανάπτυξη έχει γίνει για τηλέφωνα που τρέχουν το λειτουργικό σύστημα Android και για αυτό το σκοπό έχει μελετηθεί αυτό το λειτουργικό σύστημα.

Αρχικά γίνεται αναφορά στο Android μελετώντας τα βασικά χαρακτηριστικά του και την ιστορία του. Στην συνέχεια ακολουθεί ανάλυση των δομικών στοιχείων του λειτουργικού συστήματος και γίνεται σύγκριση με άλλα αντίστοιχα συστήματα. Γίνεται επίσης μια σύντομη αναφορά στις εκδόσεις του Android που έχουν κυκλοφορήσει μέχρι σήμερα. Επιπλέον παρουσιάζονται τα εργαλεία που χρησιμοποιήθηκαν και εξηγείται η διαδικασία της υλοποίησης που ακολουθήθηκε καθώς και η λειτουργία της σχεσιακής βάσης δεδομένων που χρησιμοποιείται στην εφαρμογή.

Για το σκοπό της σχεδίασης του συστήματος αυτού, πραγματοποιήθηκε μελέτη της πραγματικής ζωής στον τρόπο λειτουργίας των συνεργείων και αντιπροσωπειών, μελέτη περιπτώσεων βλάβης και των πληροφορίων που καταχωρούνται στο βιβλίο επισκευών του κάθε οχήματος, ώστε να είναι γνωστό τι δεδομένα πρέπει να αποθηκεύονται στην βάση δεδομένων της εφαρμογής. Παράλληλα έχουν διεξαχθεί κατάλληλες συνεντεύξεις προσώπων που έχουν τις απαραίτητες γνώσεις και βοηθητική έρευνα στο διαδίκτυο. Επιπροσθέτως έχουν μελετηθεί εγχειρίδια χρήσης πραγματικών μοντέλων αυτοκινήτων ώστε να υπάρχει πρόσβαση στα διαστήματα που πρέπει να γίνονται επισκευές.

Τέλος παρουσιάζεται η εφαρμογή που δημιουργήθηκε με το πρόγραμμα Android Studio IDE με χρήση του Software Development Kit (SDK) μαζί με έναν οδηγό χρήσης για την πλήρη εκμετάλλευσή της.

Λέξεις κλειδιά: Προγραμματισμός, Λειτουργικό Σύστημα, Android, Σχεσιακή Βάση Δεδομένων, Κινητά Τηλέφωνα.

Κεφάλαιο 1: Πρόβλημα συντήρησης αυτοκινήτων

Το κεφάλαιο αυτό περιγράφει την παρούσα κατάσταση σχετικά με τον τρόπο που πραγματοποιείται η συντήρηση των αυτοκινήτων και τα προβλήματα που δημιουργούνται στην διαδικασία αυτή. Παράλληλα αναλύονται πιθανές λύσεις αυτών των προβλημάτων καθώς και η προτεινόμενη λύση που επικράτησε στα πλαίσια της διπλωματικής εργασίας.

1.1 Ορισμός προβλήματος

Η συντήρηση των αυτοκινήτων και η καταγραφή ιστορικού των εργασιών που πραγματοποιούνται στο κάθε όχημα γίνεται με συμβατικό τρόπο μέχρι σήμερα. Σε κάθε επίσκεψη στο συνεργείο για την ολοκλήρωση εργασιών συντήρησης ή επισκευής βλάβης, συμπληρώνονται από τον χρήστη σε βιβλίο επισκευών οι εργασίες που πραγματοποιήθηκαν και οι πληροφορίες που αφορούν την τελευταία επίσκεψη στο συνεργείο. Ο σκοπός της διαδικασίας αυτής είναι να είναι προσβάσιμες αυτές οι πληροφορίες από τον μηχανικό αυτοκινήτων κατά την επίσκεψη, ώστε να είναι πιο άμεση, ορθότερη και πιο αποδοτική η διάγνωση βλαβών και ταυτόχρονα να είναι γνωστές οι απαραίτητες εργασίες συντήρησης που πρέπει να πραγματοποιηθούν, χωρίς περιττούς ελέγχους στο όχημα.

Χρησιμοποιώντας το βιβλίο επισκευών παρατηρούνται αρκετά προβλήματα. Είναι πολύ πιθανό ο ιδιοκτήτης ενός οχήματος να αμελήσει την παρακολούθηση των χιλιομέτρων στο οδόμετρο ή του διαστήματος από την προηγούμενη επίσκεψη με αποτέλεσμα, να γίνεται οδήγηση με αυξημένο κίνδυνο πρόκλησης σοβαρής βλάβης στο αυτοκίνητο ή την πρόκληση τροχαίου ατυχήματος και τραυματισμό ή θάνατο ανθρώπων. Παράλληλα, είναι πολύ πιθανό να είναι ελλιπής η καταγραφή των πληροφορίων και έτσι να μην είναι γνωστό στον οδηγό ποιες εργασίες πρέπει να πραγματοποιηθούν ή ποιο είναι το σωστό διάστημα που πρέπει να περάσει για να γίνουν. Αυτό έχει ως συνέπεια τον κίνδυνο κυκλοφορίας οχήματος με σημαντικές φθορές και δυσλειτουργίες.

Το βιβλίο επισκευών είναι απαραίτητο για την ορθή λειτουργία των αυτοκινήτων και θα υπήρχαν πολύ σοβαρά προβλήματα χωρίς αυτό. Παρόλα αυτά, απαιτείται από τον ιδιοκτήτη διαρκή παρακολούθηση των χιλιομέτρων που διανύει, του χρόνου που μεσολαβεί και την συχνότητα των επισκέψεων στο συνεργείο που ολοκληρώνονται οι εργασίες συντήρησης ή επισκευής. Επιπλέον είναι σημαντική η γνώση των σωστών διαστημάτων στα οποία πρέπει να γίνονται οι έλεγχοι και η συντήρηση (για κάθε διαφορετικό τμήμα του οχήματος) με βάση τις προδιαγραφές που έχει ορίσει ο κατασκευαστής για κάθε μοντέλο αυτοκινήτου. Τέλος, αν τα εξαρτήματα που το απαρτίζουν δεν είναι γνήσια της αντιπροσωπείας, ενδέχεται να αλλάζει η διάρκεια αντοχής τους.

1.2 Προτεινόμενη λύση

Για την προσέγγιση και την επίλυση του παραπάνω προβλήματος πρέπει να ληφθούν υπόψη τα εξής θέματα: τα χιλιόμετρα του οχήματος, τα χρονικά διαστήματα από τις επισκέψεις στο συνεργείο, το ιστορικό των επισκέψεων και οι εργασίες που πραγματοποιούνται. Οι δύο τελευταίοι παράγοντες καλύπτονται από τον συμβατικό τρόπο διαχείρισης του ιστορικού με βιβλίο επισκευών. Τα χιλιόμετρα που διανύει το όχημα και τα διαστήματα των επισκέψεων απαιτούν έναν αυτοματοποιημένο τρόπο διαχείρισης.

Πιθανή λύση στο πρόβλημα θα αποτελούσε η καταγραφή του ιστορικού από τις αντιπροσωπείες αυτοκινήτων για κάθε επίσκεψη σε μια βάση δεδομένων.

Πλεονεκτήματα	**Μειονεκτήματα**
Άμεση καταγραφή των απαραίτητων στοιχείων χωρίς την μεσολάβηση του ιδιοκτήτη του οχήματος.	Ο ιδιοκτήτης δεν έχει άμεση πρόσβαση στις πληροφορίες και απαιτείται πρόσβαση στο σύστημα της αντιπροσωπείας.
Ορθές πληροφορίες για κάθε επίσκεψη.	Ο ιδιοκτήτης θα είναι υποχρεωμένος να πραγματοποιεί την συντήρηση μόνο σε επίσημες αντιπροσωπίες.
Εύκολη πρόσβαση από τον μηχανικό που αναλαμβάνει το όχημα.	Αν υπάρξει ανάγκη για επίσκεψη σε άλλο συνεργείο δεν θα είναι γνωστό το ιστορικό του οχήματος.
	Μεγαλύτερο το κόστος της συντήρησης του οχήματος αποκλειστικά σε επίσημες αντιπροσωπίες.
	Ο χρήστης δεν γνωρίζει ποιες εργασίες πρέπει να ακολουθήσουν εκ των προτέρων.
	Υπάρχει κίνδυνος να γίνει εκπρόθεσμα η επίσκεψη στο συνεργείο.
	Υπάρχει κίνδυνος οι εργασίες να γίνουν σε λάθος διαστήματα ή και καθυστερημένα.
	Ο ιδιοκτήτης δεν ειδοποιείται αυτόματα με κάποιον τρόπο και πρέπει να παρακολουθεί τα χιλιόμετρα και τα χρονικά διαστήματα.

Πίνακας 1.1: Πλεονεκτήματα και μειονεκτήματα πρώτης πιθανής λύσης

Ομοίως με την παραπάνω λύση θα ήταν ένας πιθανός τρόπος προσέγγισης του προβλήματος, με αυτοματοποιημένο τρόπο, τα αυτοκίνητα να έχουν την δυνατότητα καταγραφής των εργασιών που ολοκληρώνονται. Παράλληλα να ειδοποιείται ο χρήστης με ένδειξη στο καντράν ακολουθούμενη από έναν χαρακτηριστικό ήχο ειδοποίησης. Σε ορισμένα καινούρια μοντέλα αυτοκινήτων υπάρχει παραπλήσια δυνατότητα σαν αυτή.

Πλεονεκτήματα	**Μειονεκτήματα**
Άμεση καταγραφή των απαραίτητων στοιχείων χωρίς την μεσολάβηση του ιδιοκτήτη του οχήματος.	Ο μηχανικός δεν έχει άμεση πρόσβαση στις πληροφορίες και απαιτείται πρόσβαση στο σύστημα του οχήματος.
Ορθές πληροφορίες για κάθε επίσκεψη.	Το όχημα θα πρέπει να είναι πιο καινούριο ώστε να υποστηρίζει την δυνατότητα καταγραφής του ιστορικού.
Εύκολη πρόσβαση από τον μηχανικό που	Υπάρχει ο κίνδυνος οι εργασίες να γίνουν σε

αναλαμβάνει το όχημα.	λάθος διαστήματα ή και καθυστερημένα.
Ειδοποίηση του ιδιοκτήτη για την επόμενη επίσκεψη μόλις εκκινεί το αυτοκίνητο.	Αν οι εργασίες πραγματοποιηθούν πριν την ειδοποίηση του οχήματος, τότε ο χρήστης δεν θα είναι ενήμερος για την επόμενη επίσκεψη την σωστή χρονική στιγμή.
	Ο χρήστης δεν γνωρίζει ποιες εργασίες πρέπει να ακολουθήσουν πριν την ειδοποίηση για την επόμενη επίσκεψη.
	Υπάρχει κίνδυνος να γίνει εκπρόθεσμα η επίσκεψη στο συνεργείο.

Πίνακας 1.2: Πλεονεκτήματα και μειονεκτήματα δεύτερης πιθανής λύσης

Τέλος μια πιθανή λύση με τα περισσότερα πλεονεκτήματα θα ήταν η δημιουργία μιας εφαρμογής για έξυπνα με επιπλέον δυνατότητες. Εκτός από το ιστορικό, θα ήταν χρήσιμο ο χρήστης να έχει τις παρακάτω δυνατότητες: να βλέπει επιλογές συνεργείων ή αντιπροσωπειών αντιστοίχως για το όχημα του, να έχει την δυνατότητα προβολής των συνεργείων στον χάρτη και να έχει περισσότερες επιλογές για τις ειδοποιήσεις.

Η μορφή της εφαρμογής θα μπορούσε να είναι διαδικτυακή ιστοσελίδα ή πρόγραμμα στον υπολογιστή. Με τον τρόπο αυτό όμως δεν είναι δυνατή η πρόσβαση των χρηστών στην εφαρμογή όταν είναι στο συνεργείο, οπότε χάνεται η λειτουργικότητα που θέλουμε να επιτύχουμε.

Βέλτιστη μορφή είναι να δημιουργηθεί μια εφαρμογή για έξυπνα κινητά με android, ώστε οι χρήστες κάθε στιγμή να έχουν πρόσβαση στο ιστορικό των επισκέψεων και να είναι δυνατή η λήψη ειδοποιήσεων για τις επερχόμενες επισκέψεις.

Πλεονεκτήματα	**Μειονεκτήματα**
Άμεση καταγραφή των απαραίτητων στοιχείων.	Ο μηχανικός δεν έχει άμεση πρόσβαση στις πληροφορίες και απαιτείται συνεννόηση με τον ιδιοκτήτη.
Ορθές πληροφορίες για κάθε επίσκεψη.	Η εφαρμογή είναι διαδικτυακή και χρειάζεται πρόσβαση στο διαδίκτυο από το κινητό τηλέφωνο.
Εύκολη πρόσβαση από τον ιδιοκτήτη και τον μηχανικό που αναλαμβάνει το όχημα.	
Ειδοποίηση του ιδιοκτήτη για την επόμενη επίσκεψη. Είναι δυνατό ο χρήστης να έχει πρόσβαση και να βλέπει τις επερχόμενες ειδοποιήσεις.	
Οι ειδοποιήσεις ορίζονται αυτόματα από την εφαρμογή με βάση την ημερομηνία της παρούσας επίσκεψης που εισάγει ο χρήστης και του χρονικού διαστήματος που ορίζει ο	

κατασκευαστής για κάθε εργασία.	
Σωστά διαστήματα επισκέψεων και χιλιομέτρων. Οι πληροφορίες συλλέγονται από τις κατασκευάστριες εταιρίες κάθε οχήματος.	

Πίνακας 1.3: Πλεονεκτήματα και μειονεκτήματα τρίτης πιθανής λύσης

Κεφάλαιο 2: Λειτουργικό σύστημα Android

Το κεφάλαιο αυτό εξηγεί λεπτομερώς τις έννοιες που κρύβονται πίσω από το Android ξεκινώντας από τον ορισμό του λειτουργικού συστήματος Android, κάνοντας μια σύντομη ιστορική αναδρομή στις εκδόσεις που έχουν κυκλοφορήσει. Στην συνέχεια περιγράφεται η αρχιτεκτονική του και τέλος γίνεται μια σύντομη αναφορά στα βασικά στοιχεία που αποτελούν τις εφαρμογές για κινητά τηλέφωνα με Android.

2.1 Εισαγωγή στο Android

Εικόνα 2.1: Το σήμα του Android.

2.1.1 Το λειτουργικό σύστημα Android και οι χρήσεις του

Λειτουργικό σύστημα (αγγλικά: Operating System) είναι το λογισμικό που σκοπός του είναι να συντονίσει και να διαχειριστεί τις διεργασίες που εκτελούνται σε έναν ηλεκτρονικό υπολογιστή και να κατανέμει τους διαθέσιμους πόρους στις εργασίες που εκτελούνται σε αυτόν. Το λειτουργικό είναι ένα μια διεπαφή επικοινωνίας μεταξύ των εφαρμογών και του υπολογιστικού συστήματος. [1] Σύμφωνα με τον J. Paul Cardle το Android είναι ένα λειτουργικό σύστημα ανοιχτού κώδικα βασισμένο στο Linux παρέχοντας έτσι εκτεταμένη ασφάλεια, αρθρωτότητα και παραγωγικότητα. «Ο πυρήνας του Android είναι μια διακλάδωση του πυρήνα Linux και έχει περαιτέρω αλλαγές αρχιτεκτονικής εκτός του τυπικού κύκλου ανάπτυξης του πυρήνα Linux». Ανοιχτός κώδικας σημαίνει ότι ο πηγαίος κώδικας ενός προγράμματος μπορεί να τροποποιηθεί και να γίνει αναδιανομή στο ευρύ κοινό από τους προγραμματιστές ελεύθερα. Ο πηγαίος κώδικάς του Android, καλείται Android Open Source Project (AOSP), ο οποίος είναι κατά κύριο λόγο αδειοδοτημένος με την άδεια Apache.

Το Android τρέχει σε φορητές συσκευές με οθόνη αφής κυρίως στα έξυπνα τηλέφωνα και ταμπλέτες αλλά χρησιμοποιείται και σε άλλα συστήματα. Τέτοια συστήματα υπάρχουν στα σύγχρονα αυτοκίνητα, τα infotainment systems (συστήματα πληροφόρησης και ψυχαγωγίας) γνωστά ως Android Auto. Άλλες χρήσεις περιλαμβάνουν σε καινούριες τηλεοράσεις που το έχουν εγκατεστημένο, γνωστό ως Android TV, και σε έξυπνα ρολόγια

γνωστό ως Android Wear. Επίσης έχει χρησιμοποιηθεί και σε ηλεκτρονικούς υπολογιστές, σε διάφορες κονσόλες ηλεκτρονικών παιχνιδιών και σε ψηφιακές φωτογραφικές μηχανές. Το Android καλύπτει μια πολύ μεγάλη γκάμα επιλογών και διαφόρων συσκευών σε πολλούς τομείς της τεχνολογίας. [4] [5]

2.1.2 Η ιστορία του Android

Αρχικά η δημιουργία του Android έγινε το 2003 από τους Andy Rubin, Rich Miner, Nick Sears και τον Chris White στο Palo Alto στην Καλιφόρνια. Το λογισμικό αυτό αρχικά προορίζονταν για ψηφιακές φωτογραφικές μηχανές αλλά τα σχέδια των δημιουργών του άλλαξαν, διότι εκείνη την περίοδο δεν ήταν μεγάλη η ζήτησή τους και έτσι επικεντρώθηκαν στα κινητά τηλέφωνα. Το 2005 η εταιρία Android Inc εξαγοράστηκε από την Google και η ανάπτυξη και διερεύνηση του Android συνέχισε μέχρι το 2007. Ο οργανισμός Open Headset Alliance (OHA) είναι υπεύθυνος για την δημιουργία, διατήρηση και ανάπτυξη του Android. Ο (OHA) δημιουργήθηκε το 2007 με αρχικό μέλος την Google και είναι μια κοινοπραξία από 48 εταιρίες τηλεπικοινωνιών, υλισμικού και λογισμικού. [6][5]

Η πρώτη συσκευή με λειτουργικό Android έγινε εμπορικά διαθέσιμη το φθινόπωρο του 2008, η οποία είναι ένα έξυπνο κινητό τηλέφωνο, και την κατασκεύασε η εταιρία ηλεκτρονικών ειδών HTC (μέλος του οργανισμού OHA). Από το 2008 ακολούθησαν πολλές αναβαθμίσεις στο Android, οι οποίες το βελτίωσαν και το έκαναν πιο σταθερό διορθώνοντας αστοχίες που υπήρχαν, προσθέτοντας παράλληλα καινούριες λειτουργίες για μεγιστοποίηση της χρησιμότητας του. [4] Τα παραπάνω σε συνδυασμό με την ελευθερία που δίνει στον χρήστη να το παραμετροποιήσει και η ευελιξία που παρέχει, το έχουν καταστήσει μια από τις πιο δημοφιλείς πλατφόρμες λογισμικού για κινητά τηλέφωνα που είναι εγκατεστημένη στα περισσότερα σύγχρονα έξυπνα τηλέφωνα της αγοράς.

2.1.3 Αναβαθμίσεις λογισμικού

Ακολουθούν στον παρακάτω πίνακα όλες οι εκδόσεις που έχουν υπάρξει μέχρι σήμερα.

Όνομα	Αριθμός έκδοσης	Ημερομηνία πρώτης σταθερής έκδοσης	Υποστηριζόμενες διορθώσεις ασφάλειας	API νούμερο
Χωρίς επίσημη ονομασία	1.0	23 Σεμπτεμβρίου 2008	Όχι	1
	1.1	9 Φεβρουαρίου 2009	Όχι	2
Cupcake	1.5	27 Απριλίου 2009	Όχι	3
Donut	1.6	15 Σεπτεμβρίου 2009	Όχι	4
Éclair	2.0-2.1	26 Οκτωμβρίου 2009	Όχι	5-7
Froyo	2.2-2.2.3	20 Μαΐου 2010	Όχι	8
Gingerbread	2.3-2.3.7	6 Δεκεμβρίου 2010	Όχι	9-10
Honeycomb	3.0-3.2.6	22 Φεβρουαρίου 2011	Όχι	11-13

Ice Cream Sandwitch	4.0-4.0.4	18 Οκτωμβρίου 2011	Όχι	14-15
Jelly Bean	4.1-4.3.1	9 Ιουλίου 2012	Όχι	16-18
KitKat	4.4-4.4.4	31 Οκτωμβρίου 2013	Όχι	19-20
Lollipop	5.0-5.1.1	12 Νοεμβρίου 2014	Όχι	21-22
Marshmallow	6.0-6.0.1	5 Οκτωμβρίου 2015	Όχι	23
Nougat	7.0-7.1.2	22 Αυγούστου 2016	Όχι	24-25
Oreo	8.0-8.1	21 Αυγούστου 2017	Ναι	26-27
Pie	9	6 Αυγούστου 2018	Ναι	28
Android 10	10	3 Σεπτεμβρίου 2019	Ναι	29
Android 11	11	8 Σεπτεμβρίου 2020	Ναι	30

Πίνακας 2.1: Όλες οι εκδόσεις του Android [7]

Εκδόσεις 1.0-1.1 χωρίς επίσημη ονομασία

Η διάθεση του Android στις συσκευές ξεκίνησε με την έκδοση 1.0 στις 23 Σεπτεμβρίου του 2008 χωρίς κωδική ονομασία. Το σύστημα εκείνη την εποχή είχε τις πολύ βασικές λειτουργίες σε αντίθεση με σήμερα. Ενσωματωμένες στο λειτουργικό αυτό ήταν οι πολύ γνωστές υπηρεσίες της Google (εφαρμογές όπως οι Gmail, Maps, Calendar και άλλες). [8] Είχε επιπλέον το Android Market, το οποίο επέτρεπε στους προγραμματιστές να φτιάχνουν εφαρμογές και στους χρήστες να τις εγκαθιστούν ή να τις ενημερώνουν. Αυτό κατέστησε πολύ εύχρηστες τις συσκευές που είχαν εγκατεστημένο το Android. Αυτή η έκδοση παρείχε στον χρήστη βασικές εφαρμογές για την υποστήριξη της κάμερας, αναπαραγωγή μουσικής και web browser. Στις 9 Φεβρουαρίου 2009 έγινε διαθέσιμη η έκδοση 1.1 αρχικά για το κινητό τηλέφωνο HTC Dream. Σε αυτή την αναβάθμιση διορθώθηκαν λάθη και προστέθηκαν μερικά νέα χαρακτηριστικά. [7]

Έκδοση 1.5 Cupcake

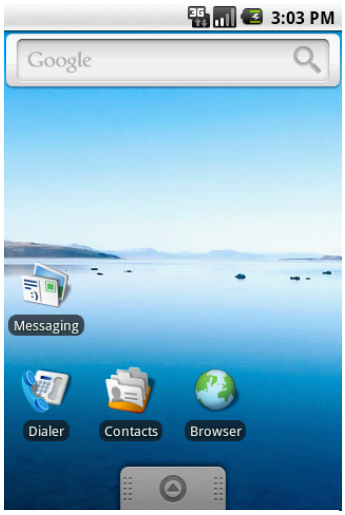
Στις 27 Απριλίου του 2009 η επόμενη αναβάθμιση Android ξεκίνησε ένα νέο σχήμα ονομάτων για κάθε νέα έκδοση (εικόνα 2.2). Μια από τις κυριότερες αλλαγές που εισήχθη ήταν ένα νέο εικονικό πληκτρολόγιο, φιλικό προς το χρήστη, με δυνατότητες πρόβλεψης κειμένου και λεξιλογίου. Μια άλλη προσθήκη ήταν η υποστήριξη των Widgets. Τα προηγούμενα είναι όψεις, από εφαρμογές, που είναι μόνιμα στην αρχική οθόνη και δείχνουν κάποιες πληροφορίες ή έχουν κάποια λειτουργία και περιοδικά ανανεώνονται για νέες πληροφορίες. [7]

Έκδοση 1.6 Donut

Η έκδοση 1.6 Donut που κυκλοφόρησε στις 15 Σεπτεμβρίου του 2009 συμπλήρωσε αρκετά κενά που υπήρχαν στην προηγούμενη. Ένα από αυτά είναι η υποστήριξη διαφόρων μεγεθών οθονών και αναλύσεων, ώστε να μπορεί να χρησιμοποιηθεί το λειτουργικό αυτό σε πολλά είδη συσκευών στο μέλλον. Επιπλέον έγινε προσθήκη υποστήριξης για δίκτυα CDMA. Τέλος άλλαξε η διεπαφή χρήστη με μικρές αισθητικές αλλαγές στο γραφικό περιβάλλον όπως

φαίνεται στην εικόνα 2.3 όπου φαίνεται στιγμιότυπο οθόνης από μια Android συσκευή. [8]

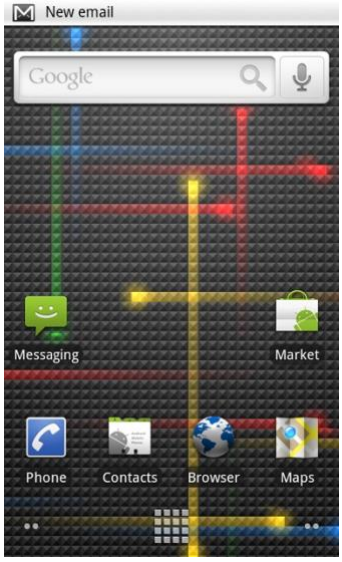
Εικόνα 2.2: Στιγμιότυπο οθόνης από την έκδοση 1.5 Cupcake.



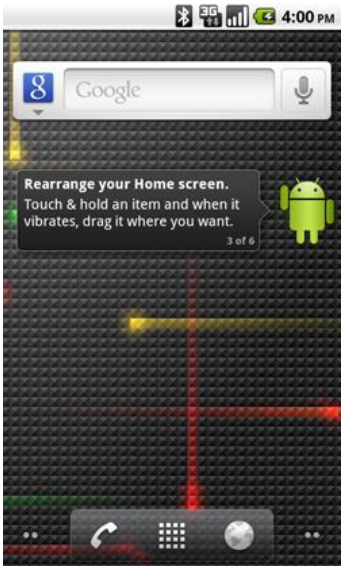
Εικόνα 2.3: Στιγμιότυπο οθόνης από την έκδοση 1.6 Donut.

Εκδόσεις 2.0-2.1 Eclair

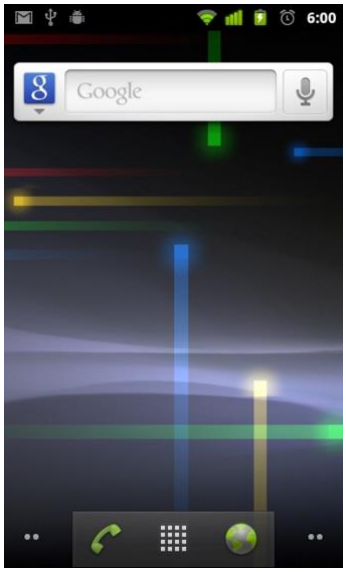
Το Eclair είναι η κωδική ονομασία που δόθηκε στις εκδόσεις 2.0-2.1 που κυκλοφόρησαν στις 26 Οκτωβρίου του 2009. Η έκδοση 2.0 είχε κάποιες μικρές αλλαγές στην διεπαφή χρήστη. Πιο συγκεκριμένα έγιναν μικρές αλλαγές στα εικονίδια, στη γραμματοσειρά, καινούριες κινούμενες ταπετσαρίες προστέθηκαν και έγινε βελτίωση στην απόκρισή της κατά τη χρήση. Προστέθηκαν επίσης καινούριες λειτουργίες όπως η δυνατότητα προσθήκης πολλαπλών λογαριασμών στην ίδια συσκευή για να συγχρονίζονται οι επαφές και τα email, υποστήριξη του Microsoft Exchange και υποστήριξη για το Bluetooth 2.1. Στην εικόνα 2.4 φαίνεται η αρχική οθόνη της έκδοσης 2.0-2.1. [9]



Εικόνα 2.4: Στιγμιότυπο οθόνης από τις εκδόσεις 2.0-2.1



Εικόνα 2.5: Στιγμιότυπο οθόνης από τις εκδόσεις 2.2-2.2.3



Εικόνα 2.6: Στιγμιότυπο οθόνης από τις εκδόσεις 2.3-2.3.7

Εκδόσεις 2.2-2.2.3 Froyo

Μερικούς μήνες αργότερα από την κυκλοφορία της έκδοσης 2.1 παρουσιάστηκε, στις 20

Μαΐου του 2010, η έκδοση 2.2 με κωδικό όνομα Froyo. Εδώ έγιναν σημαντικές τροποποιήσεις συμπεριλαμβανομένης της προσθήκης ενός “Dock” (μιας γραμμής εφαρμογών στο κάτω μέρος της αρχικής οθόνης), η πρώτη ενσάρκωση της δυνατότητας φωνητικών εντολών που επιτρέπει την εκτέλεση λειτουργιών όπως η λήψη σημειώσεων, λήψη οδηγιών από την εφαρμογή χάρτες και άλλες. Επιπλέον το πρόγραμμα περιήγησης του συστήματος υποστηρίζει το Adobe Flash που εκείνη την εποχή ήταν ευρέως διαδεδομένο. Στην εικόνα 2.5 φαίνεται η αρχική οθόνη της έκδοσης 2.2-2.2.3. [8]

Εκδόσεις 2.3-2.3.7 Gingerbread

Στις 6 Δεκεμβρίου του 2010 αναβαθμίστηκε ξανά το Android στην έκδοση 2.3. Το Gingerbread έκανε πιο απλοϊκό και ταχύτερο το περιβάλλον χρήσης. Τα χρώματα άλλαξαν σε αυτά που έχει το σύμβολο του λειτουργικού αυτού συστήματος, πιο συγκεκριμένα φωτεινό πράσινο. Παράλληλα βελτιώθηκαν τα εργαλεία διαχείρισης της μπαταρίας και των εφαρμογών που συνέβαλε στην αποδοτικότερη χρήση μιας Android συσκευής. Άρχισε να υπάρχει υποστήριξη για πολύ μεγάλες οθόνες και αναλύσεις, για VoIP διαδικτυακές κλήσεις, για NFC και άλλα χαρακτηριστικά που είναι μέρος του Android όπως το ξέρουμε σήμερα. Στην εικόνα 2.6 φαίνεται η αρχική οθόνη της έκδοσης 2.3-2.3.7. [7]

Εκδόσεις 3.0-3.2.6 Honeycomb

Στις 22 Φεβρουαρίου του 2011 ήταν η επόμενη αναβάθμιση του πολύ γνωστού πλέον λειτουργικού συστήματος, η έκδοση 3.0 Honeycomb. Εκείνη την περίοδο ήρθε στην αγορά μια καινούρια κατηγορία φορητών συσκευών οι ταμπλέτες (αγγλικά: tablet) που είχαν διαφορετικές σχεδιαστικές απαιτήσεις στο περιβάλλον χρήσης. Για το λόγο αυτό η έκδοση αυτή δεν ήταν προσανατολισμένη στα έξυπνα τηλέφωνα τόσο αλλά στην νέα αυτή κατηγορία,

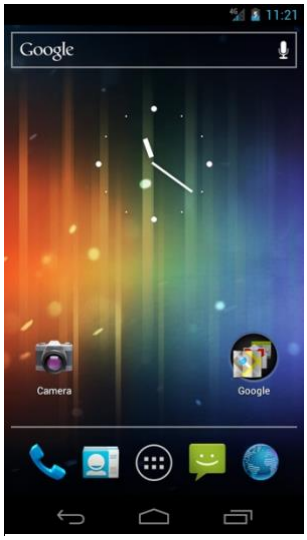
Εικόνα 2.7: Στιγμιότυπο οθόνης από τις εκδόσεις 3.0-3.2

διότι η Google ήθελε να επεκταθεί και στα tablet. Δεν ήταν τόσο επιτυχημένη αυτή η επέκταση, ωστόσο έγιναν κάποιες βελτιώσεις στο λογισμικό. «Η Honeycomb έκδοση παρουσίασε μια

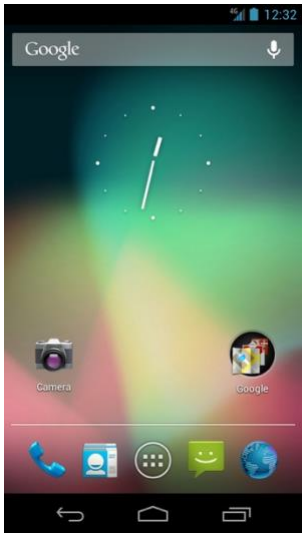
δραματικά επαναπροσδιορισμένη διεπαφή χρήστη για Android. Είχε ένα διαστημικό "ολογραφικό" σχέδιο και έδωσε έμφαση στο να αξιοποιήσει στο έπακρο το χώρο της οθόνης ενός tablet». Ήταν η αρχή του τέλους για τα φυσικά κουμπιά στις συσκευές διότι ήταν το πρώτο λογισμικό που εισήγαγε εικονικά κουμπιά στην οθόνη ως κύρια είσοδο για εντολές πλοήγησης στο σύστημα. Επίσης εισήχθη ο νέος τρόπος με τον οποίο εμφανίζονται οι εφαρμογές που τρέχουν στο παρασκήνιο στην οθόνη (σε μορφή καρτελών). Η εικόνα 2.7 παραπάνω δείχνει τις νέες αυτές αλλαγές. [8]

Εκδόσεις 4.0-4.0.4 Ice Cream Sandwich

Στις 18 Οκτωμβρίου του 2011 το Android ήταν η πιο μεγάλη αναβάθμιση που είχε δεχτεί μέχρι τότε, πολλά όμως χαρακτηριστικά του ξεκίνησαν από την προηγούμενη έκδοση συμπεριλαμβανομένων των εικονικών πλήκτρων πλοήγησης, οι χρωματικές αποχρώσεις στην διεπαφή χρήστη, η καλύτερη υποστήριξη για Widget και η απεικόνιση των πολλαπλών εργασιών με λίστα μικρογραφιών των εφαρμογών. Επιπλέον η γραμματοσειρά του συστήματος άλλαξε και η νέα ήταν ειδικά διαμορφωμένη ώστε να ταιριάζει καλύτερα σε οθόνες με μεγαλύτερη ανάλυση. Ο αντιπρόεδρος σχεδιασμού της Google, Matias Duarte, σημείωσε ότι η παλιά γραμματοσειρά «αγωνίστηκε να επιτύχει τόσο το άνοιγμα όσο και την πυκνότητα πληροφοριών που θέλαμε στο Ice Cream Sandwich». Ταυτόχρονα ένα από τα



Εικόνα 2.8: Στιγμιότυπο οθόνης από την έκδοση 4.0



Εικόνα 2.9: Στιγμιότυπο οθόνης από την έκδοση 4.1

Εικόνα 2.10: Στιγμιότυπο οθόνης από την έκδοση 4.4

καλύτερα υλοποιημένα χαρακτηριστικά του λειτουργικού που άρχισε να είναι ξεπερασμένο ανανεώθηκε, η οθόνη ειδοποιήσεων απέκτησε περισσότερες επιλογές για την διαχείριση των ειδοποιήσεων. Επίσης αλλαγές έγιναν και στο εικονικό πληκτρολόγιο που με κάθε έκδοση επιδέχεται μικρές βελτιώσεις όπως και σε αυτή. Ένα αξιοσημείωτο νέο χαρακτηριστικό που προστέθηκε, εκτός από το κλείδωμα με κωδικό ή μοτίβο που υποστηρίζονταν από πριν, είναι το ξεκλείδωμα προσώπου (αγγλικά: face unlock) που χρησιμοποιεί την μπροστινή κάμερα της συσκευής και αν υπάρχει αντιστοίχιση με το πρόσωπο του κατόχου τότε έχει πρόσβαση ο χρήστης. Η εικόνα 2.8 παραπάνω δείχνει αποτύπωμα οθόνης αυτής της έκδοσης Android. [10]

Εκδόσεις 4.1-4.3.1 Jelly Bean

Στις 27 Ιουλίου του 2012 ανακοινώθηκε η πρώτη έκδοση Jelly Bean και έφερε μαζί της ακόμα πιο ομαλή διεπαφή. Σε συνδυασμό με το ρυθμό ανανέωσης της οθόνης έχουμε το αποτέλεσμα της εικόνας 2.9 που είναι μεγάλη βελτίωση. Πρόσθετα, η Jelly Bean έκδοση μας έδωσε μια πρώτη γεύση από το Google Now, το θεαματικό βοηθητικό πρόγραμμα πρόβλεψης

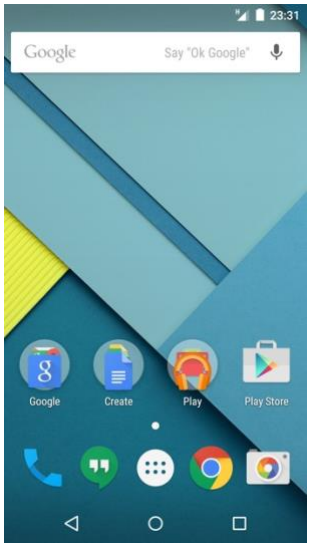
νοημοσύνης. Έγινε προσθήκη επεκτάσιμων και διαδραστικών ειδοποιήσεων και ενός διευρυμένου συστήματος φωνητικής αναζήτησης που προάχθηκε σε πιο προηγμένο σύστημα για την εμφάνιση αποτελεσμάτων αναζήτησης, με έμφαση στα αποτελέσματα βάσει καρτών που προσπάθησαν να απαντήσουν απευθείας σε ερωτήσεις. Στα tablet προστέθηκε η δυνατότητα πολλαπλών χρηστών και την πρώτη του εμφάνιση έκανε στο Android ένας πίνακας γρήγορων ρυθμίσεων. Επιπροσθέτως, τα Widget απέκτησαν μεγαλύτερη ευελιξία, είναι πιο εύκολο από ποτέ να χωρέσουν πολλά σε μια αρχική οθόνη χωρίς κόπο από τον χρήστη. Η εικόνα 2.9 παραπάνω δείχνει αποτύπωμα οθόνης αυτής της έκδοσης Android. [8]

Εκδόσεις 4.4-4.4.4 KitKat

Η Google ανακοίνωσε την έκδοση KitKat τον Οκτώβριο του 2013. Αρχικά προοριζόταν να έχει το όνομα Key Lime Pie αλλά λίγοι άνθρωποι γνώριζαν την γεύση αυτού του γλυκίσματος και έτσι άλλαξε. Για μια ακόμα φορά έγιναν πολλές οπτικές αλλαγές στην διεπαφή χρήστη από την έκδοση 4.0 παρόλο που δεν ήταν κύρια ενημέρωση. Οι μεγαλύτερες αλλαγές έγιναν στην αρχική οθόνη η οποία απέκτησε εικονικά κουμπιά και μπάρα ειδοποιήσεων με διαφάνεια. Το συρτάρι εφαρμογών (αγγλικά: app drawer) ανανεώθηκε επίσης και έγινε προσβάσιμο απευθείας σε αυτή την οθόνη. «Το Android 4.4 είδε επίσης την πρώτη έκδοση της λειτουργίας "OK, Google" αλλά εδώ, η ενεργοποίηση ανοιχτής ακρόασης λειτουργεί μόνο όταν η οθόνη είναι ήδη ενεργοποιημένη και βρίσκεται ο χρήστης είτε στην αρχική οθόνη είτε μέσα στην εφαρμογή Google». Η εικόνα 2.10 παραπάνω δείχνει στιγμιότυπο οθόνης της έκδοσης 4.4 KitKat. [8]

Εκδόσεις 5.0-5.1.1 Lollipop

Τρία χρόνια μετά την κυκλοφορία του 4.0 Ice Cream Sandwitch, στις 12 Νοεμβρίου του 2014, ήρθε το 5.0 Lollipo που έφερε μια νέα σχεδιαστική λογική την “Material Design” η οποία, άλλαξε την εικόνα και την αίσθηση των εφαρμογών σε ολόκληρο το σύστημα



Εικόνα 2.11: Στιγμιότυπο οθόνης από τις εκδόσεις 5.0, 6.0 και 7.0. Στις εκδόσεις 6.0 και 7.0 αλλάζουν πολύ λίγα στοιχεία και ένα από αυτά είναι το φόντο στην αρχική οθόνη.

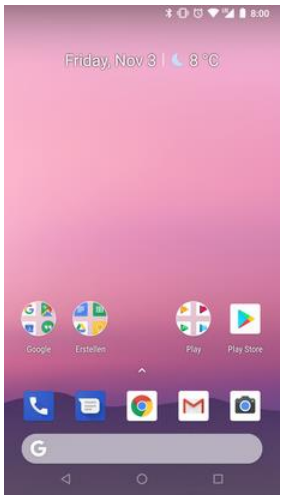
συμπεριλαμβανομένων των εφαρμογών Gmail, YouTube, Google Maps καθώς και οι προ εγκατεστημένες εφαρμογές όπως το τηλέφωνο και το ημερολόγιο. Η ιδέα βασίζεται σε καρτέλες και είχε διασκορπιστεί σε όλο το Android. Έγινε ένα βασικό μοτίβο διεπαφής χρήστη

που θα επηρέαζε την εμφάνιση όλων των στοιχείων του συστήματος από τις ειδοποιήσεις, οι οποίες τώρα εμφανίστηκαν στην οθόνη κλειδώματος για άμεση πρόσβαση, μέχρι την λίστα με τις πρόσφατες εφαρμογές. Το Lollipop εισήγαγε πολλά νέα χαρακτηριστικά στο Android, συμπεριλαμβανομένου του φωνητικού ελέγχου “hands-free” μέσω της εντολής "OK, Google", την υποστήριξη για πολλούς χρήστες σε τηλέφωνα και λειτουργία προτεραιότητας για καλύτερη διαχείριση ειδοποιήσεων. Επίσης η Google έχει επανασχεδιάσει τον τρόπο λειτουργίας των πολλαπλών εργασιών και έγινε εισαγωγή ενός νέου συστήματος που βελτίωσε την απόδοση της μπαταρίας το “Project Volta”. Παρόλα αυτά η έκδοση είχε κάποιες αστοχίες και για το λόγω αυτό, αναβαθμίστηκε στην έκδοση 5.1. Η εικόνα 2.11 δείχνει στιγμιότυπο οθόνης της έκδοσης 5.0 Lollipop. [8] [10]

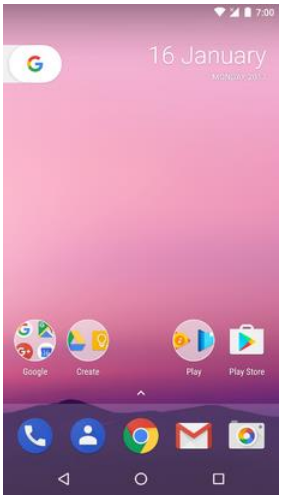
Εκδόσεις 6.0.-6.0.1 Marshmallow

Στις 5 Οκτωμβρίου του 2015 ανακοινώθηκε η έκδοση 6.0 Marshmallow η οποία δεν είχε πολλά νέα κύρια χαρακτηριστικά, αλλά έγιναν πολλές κρυφές αλλαγές στο όλο σύστημα που οδήγησε στην ωρίμανση του λειτουργικού συστήματος. Το πιο ελκυστικό στοιχείο του Marshmallow ήταν μια λειτουργία αναζήτησης οθόνης που ονομάζεται “Now On Tap”, κάτι που είχε δυναμικό, αλλά δεν είχε αξιοποιηθεί πλήρως. Έτσι τον επόμενο χρόνο αποσύρθηκε από την επόμενη έκδοση. Άλλες σημαντικές προσθήκες είναι το Android Pay που επιτρέπει στους χρήστες να κάνουν πληρωμές από το κινητό τους τηλέφωνο μέσω NFC, η δυνατότητα να επιλέγει ο χρήστης τις άδειες των εφαρμογών, δηλαδή τι πόρους μπορεί να χρησιμοποιήσει μια εφαρμογή για παράδειγμα αν επιτρέπεται να χρησιμοποιήσει την κάμερα και να έχει πρόσβαση σε αρχεία της συσκευής. Τέλος, η διαχείριση των εφαρμογών στο παρασκήνιο, από το λειτουργικό σύστημα, έγινε εξυπνότερη με την προσθήκη της λειτουργίας “Doze” η οποία βελτίωσε την απόδοση των συσκευών Android. Το συρτάρι εφαρμογών στην αρχική οθόνη επίσης βελτιώθηκε. Η εικόνα 2.11 παραπάνω δείχνει στιγμιότυπο οθόνης της έκδοσης 6.0 Marshmallow. [8] [10]

Εικόνα 2.12: Στιγμιότυπο οθόνης από την έκδοση 7.1



Εικόνα 2.13: Στιγμιότυπο οθόνης από την έκδοση 8.0



Εικόνα 2.14: Στιγμιότυπο οθόνης από την έκδοση 8.1

Εκδόσεις 7.0-7.1.2 Nougat

Το Android 7.0 Nougat κυκλοφόρησε επίσημα στις 22 Αυγούστου του 2016. Σε αυτή την αναβάθμιση λήφθηκαν υπόψη τα έξυπνα τηλέφωνα με μεγάλη οθόνη και έτσι η λειτουργία “split-screen-multitasking” έγινε πραγματικότητα. Η Google έδωσε στους χρήστες την

δυνατότητα να χρησιμοποιούν δύο εφαρμογές ταυτόχρονα. Μαζί με αυτή την λειτουργία οι χρήστες μπορούν με διπλό άγγιγμα στο κουμπί πρόσφατων εφαρμογών να «μεταπηδήσουν» μεταξύ της τελευταίας εφαρμογής και της προηγούμενης που χρησιμοποιούσε ο χρήστης. Παράλληλα προστέθηκαν οι «γρήγορες απαντήσεις» (αγγλικά: quick replies) είτε για εφαρμογές συστήματος είτε για εφαρμογές τρίτων και πλέον οι γρήγορες ρυθμίσεις είναι παραμετροποιήσιμες με βάση τις προτιμήσεις του χρήστη. Ίσως το πιο σημαντικό μεταξύ των βελτιώσεων του Nougat, ήταν η κυκλοφορία του Βοηθού Google. Αυτή η λειτουργία συνόδευσε την ανακοίνωση του πρώτου πλήρως κατασκευασμένου τηλεφώνου της Google, το Pixel, περίπου δύο μήνες μετά το ντεμπούτο του Nougat. Η εικόνα 2.11 παραπάνω δείχνει στιγμιότυπο οθόνης της έκδοσης 7.0 και η εικόνα 2.12 δείχνει στιγμιότυπο οθόνης από την έκδοση 7.1 που εμπεριέχει κάποιες σχεδιαστικές αλλαγές. [8] [10]

Εκδόσεις 8.0-8.1 Oreo

Το Android Oreo που κυκλοφόρησε στις 21 Αυγούστου του 2017 πρόσθεσε μια ποικιλία σημαντικών αλλαγών στην πλατφόρμα, όπως τη λειτουργία εικόνας σε εικόνα (αγγλικά: picture-in-picture), την επιλογή αναβολής ειδοποιήσεων και κανάλια ειδοποιήσεων που προσφέρουν καλό έλεγχο του τρόπου με τον οποίο οι εφαρμογές μπορούν να ειδοποιήσουν τον χρήστη. Το Android TV έλαβε μια υπολογίσιμη αναβάθμιση και για πρώτη φορά το Android εντάσσεται στην λογική του Project Treble, μια αρχιτεκτονική που δίνει την δυνατότητα να είναι πιο εύκολες και γρήγορες οι αναβαθμίσεις για τους κατασκευαστές του υλικού. Μέχρι τώρα στα έξυπνα τηλέφωνα πολλών εταιριών οι καινούριες αναβαθμίσεις του λογισμικού καθυστερούσαν σημαντικά και έτσι οι νέες εκδόσεις του λογισμικού είναι πιο προσβάσιμες στους χρήστες από ποτέ. Η εικόνα 2.13 παραπάνω δείχνει στιγμιότυπο οθόνης της έκδοσης 8.0 και η εικόνα 2.14 δείχνει στιγμιότυπο οθόνης από την έκδοση 8.1 που εμπεριέχει κάποιες σχεδιαστικές αλλαγές. [8] [10]

Έκδοση 9 Pie

Στις 6 Αυγούστου του 2018 το Android αναβαθμίστηκε στην έκδοση 9.0 η οποία, επαναπροσδιόρισε τον τρόπο με τον οποίο ο χρήστης αλληλοεπιδρά με το λογισμικό και εισήγαγε τις χειρονομίες σε συνδυασμό με τα εικονικά πλήκτρα οι οποίες αντικατέστησαν στο μέλλον τα εικονικά πλήκτρα που υπήρχαν μέχρι τότε. Το κεντρικό κουμπί έγινε πολυχρηστικό επειδή απέκτησε χειρονομία για την εμφάνιση των πρόσφατων εφαρμογών σε συνδυασμό με ένα πλήκτρο για επιστροφή. Παράλληλα, η Google ανακοίνωσε την εκδοχή της για την παρακολούθηση και περιορισμό της χρήσης εφαρμογών που υπόσχεται να παρακολουθεί τις ειδοποιήσεις που λαμβάνει ο χρήστης, το χρόνο που αφιερώνεται σε εφαρμογές και πόσο συχνά ελέγχει ο χρήστης το τηλέφωνό του για να παρακολουθεί τι συμβαίνει ώστε να υπάρχει γονικός έλεγχος σε περίπτωση που ο χρήστης είναι νεαρής ηλικίας. Επιπλέον, η τεχνητή νοημοσύνη προστέθηκε σε διάφορα σημεία του λειτουργικού και έγιναν μικρές σχεδιαστικές αλλαγές στο γραφικό περιβάλλον. Η εικόνα 2.15 παρακάτω δείχνει στιγμιότυπο οθόνης της έκδοσης 9.0. [10]

Έκδοση 10

Τον Σεπτέμβριο στις 3 του μήνα του 2019 το Android προχώρησε στην έκδοση 10 του δημοφιλές λειτουργικού συστήματος η οποία, ήταν η πρώτη μετά από πολύ καιρό που έγινε γνωστή χωρίς ονομασία γλυκίσματος. Αναβαθμίστηκε ξανά ο τρόπος που ο χρήστης πλοηγείται στο σύστημα και πλέον βασίζεται σε χειρονομίες μόνο, αλλά σε αντίθεση με το Android Pie υπάρχει επιλογή να χρησιμοποιηθεί ο παραδοσιακός τρόπος δηλαδή, τα εικονικά

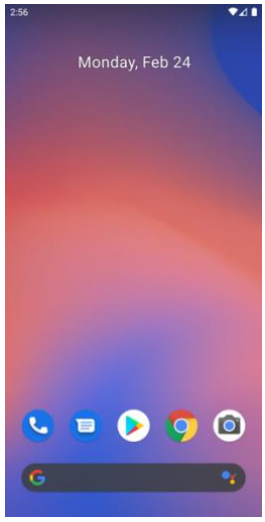
πλήκτρα στην οθόνη. Το Android 10 εισάγει ένα νέο τρόπο που οι ενημερώσεις του συστήματος γίνονται σε «στυλ επείγουσας επιδιόρθωσης» που τελικά επιτρέπουν ταχύτερες και πιο συνεπείς ενημερώσεις μικρών, στενά εστιασμένων ενημερώσεων κώδικα. Και το λογισμικό έχει πολλές άλλες σημαντικές βελτιώσεις, συμπεριλαμβανομένου ενός ενημερωμένου συστήματος δικαιωμάτων που δίνει περισσότερο έλεγχο στο χρήστη για το πώς και πότε οι εφαρμογές έχουν πρόσβαση σε δεδομένα τοποθεσίας καθώς και ένα διευρυμένο σύστημα για την προστασία μοναδικών αναγνωριστικών συσκευών. Η εικόνα 2.16 παρακάτω δείχνει στιγμιότυπο οθόνης της έκδοσης 10. [10]

Έκδοση 11

Στις 3 Σεπτεμβρίου του 2020 το Android ανανεώθηκε ξανά με την έκδοση 11. Είναι μια πολύ σημαντική αναβάθμιση, διότι οι πιο κύριες αλλαγές είναι στο κομμάτι της ιδιωτικότητας. Η Google χτίζει πάνω στο διευρυμένο σύστημα αδειών της προηγούμενης έκδοσης και προσθέτει την ιδιότητα οι χρήστες να αποφασίζουν για συγκεκριμένες άδειες σχετικές με τη πρόσβαση στη κάμερα, το μικρόφωνο και τις υπηρεσίες τοποθεσίας περιορισμένες σε μιας χρήσης βάση. Ταυτόχρονα έγινε δυσκολότερη η ενεργοποίηση αδειών υποβάθρου, ώστε να μην τις ενεργοποιήσει ο χρήστης καταλάθως. Ένα νέο χαρακτηριστικό του λειτουργικού είναι πως όταν περάσει ένα χρονικό διάστημα από την χρήση μιας εφαρμογής, πρέπει να ξανά επιτρέψει ο χρήστης τις άδειες της. Επιπλέον αυξάνονται τα στοιχεία του Android που μπορούν να είναι μεμονωμένα, σαν εφαρμογές στο PlayStore, ώστε να είναι πιο άμεση η ενημέρωση από την Google και να μην είναι απαραίτητη η εμπλοκή των εταιριών κινητής τηλεφωνίας ή των κατασκευαστών των συσκευών αυτών. «Το Android 11 έχει πολλές άλλες μικρές αλλά σημαντικές βελτιώσεις συμπεριλαμβανομένης μιας νέας ενότητας Ιστορικού ειδοποιήσεων, μιας λειτουργίας εγγενούς εγγραφής οθόνης και ενός αυτοματοποιημένου συστήματος προγραμματισμού για το Dark Theme σε όλο το σύστημα». Η εικόνα 2.17 παρακάτω δείχνει στιγμιότυπο οθόνης της έκδοσης 11. [8]

Εικόνα 2.15: Στιγμιότυπο οθόνης από την έκδοση 9

Εικόνα 2.16: Στιγμιότυπο οθόνης από την έκδοση 10

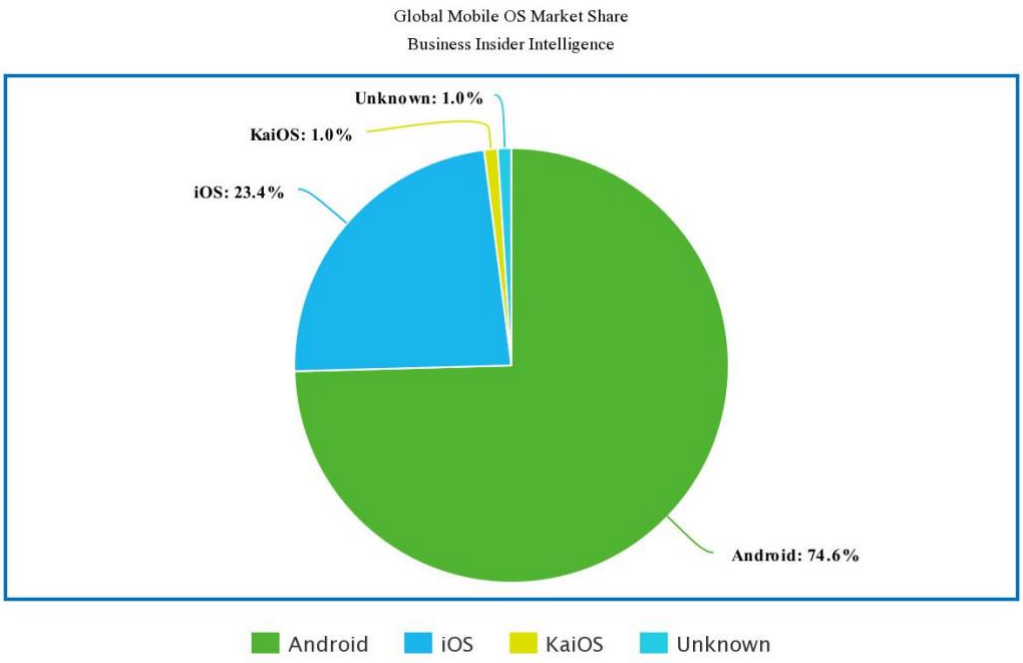


Εικόνα 2.17: Στιγμιότυπο οθόνης από την έκδοση 11

2.1.4 Εκτίμηση διαφορών με άλλα λειτουργικά συστήματα

Κατά καιρούς έχουν δημιουργηθεί πολλά λειτουργικά συστήματα για φορητές συσκευές αλλά λίγα έχουν επικρατήσει στο 2022. Αυτό συμβαίνει διότι η τεχνολογία προχωράει και έχουμε περάσει στην εποχή όπου το μεγαλύτερο μερίδιο της αγοράς καλύπτουν κυρίως τα έξυπνα

τηλέφωνα και σε πολύ μικρό βαθμό τα απλά τηλέφωνα με πλήκτρα που έχουν πιο περιορισμένες λειτουργίες. Στις μέρες μας το Android τρέχει στις περισσότερες κινητές συσκευές. Πριν γίνει δημοφιλές το Android τα πιο γνωστά λειτουργικά συστήματα για κινητά τηλέφωνα ήταν το Symbian της Nokia, το Windows Mobile της Microsoft (αργότερα μετονομάστηκε σε Windows Phone), το BlackBerry OS της BlackBerry. Πλέον αυτά τα



Εικόνα 2.18: Το μερίδιο αγοράς των πιο διαδεδομένων λειτουργικών συστημάτων για κινητά τηλέφωνα.

λειτουργικά δεν χρησιμοποιούνται από τις εταιρίες για τις καινούριες συσκευές και έχουν αντικατασταθεί σε μεγάλο βαθμό από το Android το οποίο, οι εταιρίες τεχνολογίας επιλέγουν σε μεγάλο βαθμό επειδή είναι δωρεάν, ανοιχτού κώδικα και μπορεί να χρησιμοποιηθεί σαν βάση για την δημιουργία έκδοσης του λειτουργικού αυτού με τις επιλογές που επιθυμεί η κατασκευάστρια εταιρία. Τα πιο διαδεδομένα λειτουργικά συστήματα είναι το Android της Google, το iOS της Apple και το KaiOS της KaiOS Technologies. Στην παραπάνω εικόνα το διάγραμμα δείχνει το ποσοστό των προαναφερθέντων λειτουργικών συστημάτων που έτρεχαν στις φορητές συσκευές για το έτος 2019 (εικόνα 2.18).

Android

Το Android όπως προαναφέρθηκε είναι δωρεάν και ανοιχτού κώδικα αναπτυγμένο πάνω σε Linux, γεγονός που το καθιστά προσβάσιμο από όλες τις εταιρίες που θέλουν να το χρησιμοποιήσουν ως λειτουργικό σύστημα στις συσκευές τους. Όπως φαίνεται στην εικόνα 2.18 είναι εγκατεστημένο στα περισσότερα έξυπνα τηλέφωνα της αγοράς. Έχει πολλά πλεονεκτήματα όπως το γεγονός ότι είναι πλήρως παραμετροποιήσημο, η πρόσβαση και ο πλήρης έλεγχος της συσκευής είναι διαθέσιμος και υπάρχουν πολλές επιλογές εγγενώς που δίνει το σύστημα ώστε κάθε συσκευή να ταιριάζει στις ανάγκες και προτιμήσεις του χρήστη. Επίσης είναι αρκετά εύκολο να γίνεται μεταφορά αρχείων από και προς το έξυπνο τηλέφωνο χωρίς να χρειάζεται κάποια εφαρμογή μέσω της οποίας να γίνει η μεταφορά, το Android το φροντίζει αυτό. Παράλληλα υπάρχει πληθώρα διαθέσιμων εφαρμογών (όπως και στο iOS) για εγκατάσταση και είναι πολύ εύκολο για τους προγραμματιστές να αναπτύξουν εφαρμογές, επειδή υπάρχουν εργαλεία που κάνουν την διαδικασία πολύ εύκολη και στο PlayStore δεν χρειάζεται να γίνει έγκριση των εφαρμογών για να γίνουν διαθέσιμες. Συν της άλλης δεν

απαιτείται κάποια συνδρομή από τους προγραμματιστές για να μπορούν να κάνουν διαθέσιμη την εφαρμογή τους στο PlayStore εκτός από ένα μικρό ποσό που θα πρέπει αρχικά να κατατεθεί. Επιπλέον υπάρχουν κινητά τηλέφωνα με Android για κάθε προϋπολογισμό, για το λόγο ότι υπάρχει μεγάλη ποικιλία τέτοιων συσκευών από τις οποίες μπορούν οι χρήστες να επιλέξουν.

Παρόλα αυτά, υπάρχουν κάποιες αστοχίες του λογισμικού αυτού μια από τις οποίες είναι ότι ο χρόνος που χρειάζεται για να γίνει αναβάθμιση σε μια καινούρια έκδοση λογισμικού είναι αρκετός και σημαντικές ενημερώσεις ασφαλείας μπορεί να καθυστερήσουν πολύ μέχρι να εγκατασταθούν. Ο κύκλος αναβαθμίσεων αυτών των συσκευών είναι πολύ μικρότερος σε σχέση με το iOS. Επιπλέον υπάρχουν εφαρμογές προς εγκατάσταση που δεν είναι ιδιαίτερα ποιοτικές όσο στο iOS διότι δεν υπάρχει έλεγχος. Αυτά τα χαρακτηριστικά καθιστούν το Android λιγότερο ασφαλές σε απειλές από ιούς ή παραβίαση του συστήματος.

iOS

Το iOS είναι λογισμικό για φορητές συσκευές που παρουσιάστηκε για πρώτη φορά το 2007 από την Apple Inc. Χρησιμοποιείται αποκλειστικά σε συσκευές της Apple (εικόνα 2.18) σε αντίθεση με την Google που δίνει την άδεια για εγκατάσταση του λογισμικού της σε άλλες εταιρίες. Είναι σύστημα κλειστού τύπου, δηλαδή δεν επιτρέπει στον χρήστη να παραμετροποιήσει και να ρυθμίσει την συσκευή του σε μεγάλο βαθμό εκτός από κάποιες βασικές επιλογές που παρέχει. Διαφέρει πολύ η σχεδιαστική του λογική σε σχέση με το Android και αυτό φαίνεται στην εικόνα 2.19 παρακάτω, όπου οι επιλογές για την διάταξη των εικονιδίων είναι περιοσμένη. Επιπλέον η μεταφορά των αρχείων σε αυτές τις συσκευές δεν είναι τόσο απλή διότι απαιτείται να γίνει μέσω του προγράμματος iTunes.

Όπως και στο Android οι διαθέσιμες εφαρμογές για εγκατάσταση είναι άφθονες και είναι σχετικά εύκολο να γίνει η ανάπτυξη τους, αρκεί να γίνει η κατάλληλη συνδρομή από τον

Εικόνα 2.19: Στιγμιότυπο της αρχικής οθόνης από συσκευή που τρέχει την τελευταία έκδοση του iOS (την 14).

προγραμματιστή για να μπορεί να την διαθέσει στο App Store της Apple. Σε αυτό το λειτουργικό σύστημα οι εφαρμογές συνήθως είναι πιο ποιοτικές και οι προγραμματιστές δίνουν μεγαλύτερη έμφαση από ότι σε αυτές στο Android, διότι είναι μεγαλύτερη η αμοιβή που λαμβάνουν και επίσης υπάρχει έλεγχος πριν γίνουν διαθέσιμες στο κοινό. Λόγω αυτών των περιορισμών και του τρόπου με τον οποίο έχει δημιουργηθεί το iOS η ασφάλεια και η σταθερότητα που παρέχει το λειτουργικό σύστημα είναι μεγαλύτερη από ότι στο Android. Το λογισμικό αυτό αναβαθμίζεται σε τακτά χρονικά διαστήματα, διότι η εταιρία το διανέμει στις συσκευές που παράγει. Έτσι είναι δυνατό να γίνει η διαδικασία πιο άμεσα διότι δεν μεσολαβούν άλλες εταιρίες στην διαδικασία της ανάπτυξης και έχει πολύ μεγαλύτερο κύκλο αναβαθμίσεων σε σχέση με το Android. Για παράδειγμα σε ένα καινούριο κινητό τηλέφωνο με Android ο κύκλος του λογισμικού διαρκεί περίπου τρία με τέσσερα χρόνια σε αντίθεση με ένα καινούριο κινητό τηλέφωνο με iOS που διαρκεί τουλάχιστον έξι χρόνια. Επιπλέον το iOS έχει την τάση τα τελευταία χρόνια να έχει λίγες αστοχίες οι οποίες διορθώνονται άμεσα.

KaiOS

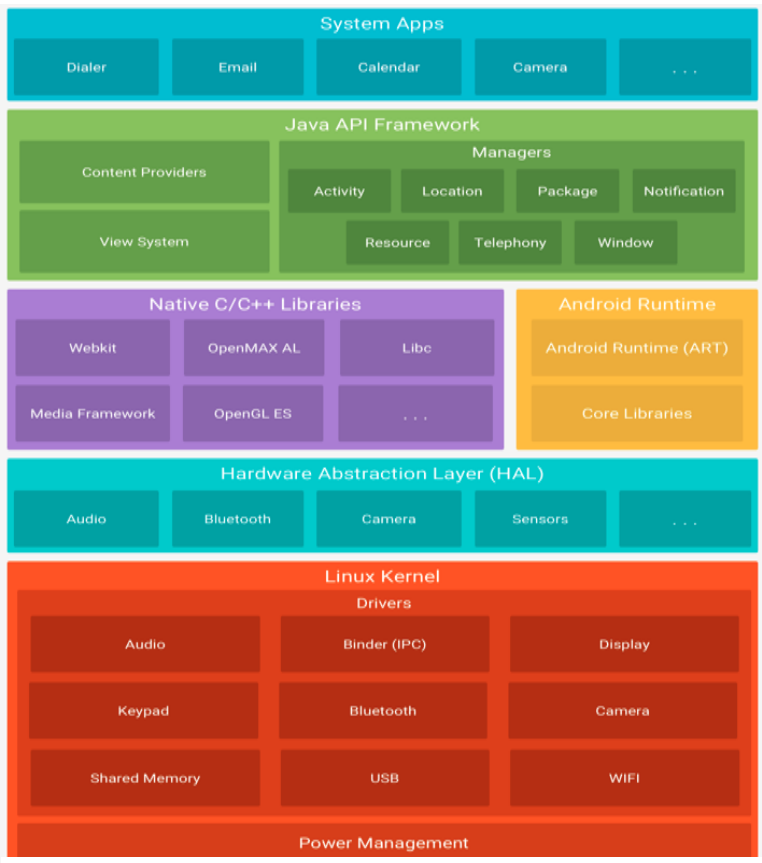
Το KaiOS είναι ένα λειτουργικό σύστημα βασισμένο σε Linux το οποίο αναπτύχθηκε από την KaiOS Technologies Limited, μια εταιρεία που εδρεύει στο Kowloon του Χονγκ Κονγκ. Χρησιμοποιείται για συσκευές με πληκτρολόγιο έχοντας μεγαλύτερο μέτοχο την κινεζική πολυεθνική εταιρεία ηλεκτρονικών ομίλων TCL Corporation. Το λογισμικό αυτό κυκλοφόρησε πρώτη φορά το 2017 και υποστηρίζει 4G LTE E, VoLTE, GPS και Wi-Fi με εφαρμογές που βασίζονται σε HTML5 προσφέροντας μεγαλύτερη διάρκεια μπαταρίας σε συσκευές χωρίς οθόνη αφής με βελτιστοποιημένη διεπαφή χρήστη, απαιτώντας λιγότερη μνήμη. Μια ειδική αγορά εφαρμογών (KaiStore) επιτρέπει στους χρήστες να κάνουν λήψη εφαρμογών, αν και ορισμένες υπηρεσίες έχουν προ φορτωθεί ως εφαρμογές HTML5, όπως το Facebook και το YouTube. Από την 1η Απριλίου του 2020, υπάρχουν πάνω από πεντακόσιες εφαρμογές στο KaiStore. Το λειτουργικό σύστημα είναι σχετικά ελαφρύ στη χρήση πόρων υλικού και είναι σε θέση να λειτουργεί σε συσκευές με μνήμη μόλις 256 MB. Το KaiOS κάνει τα παλαιού τύπου τηλέφωνα πιο έξυπνα, με περισσότερες λειτουργίες και πιο άνετα στην χρήση αλλά δεν αντικαθιστά τα έξυπνα τηλέφωνα. Είναι το τρίτο πιο γνωστό λειτουργικό σύστημα με ποσοστό 1% στην αγορά των τηλεφώνων (εικόνα 2.18). Οι εφαρμογές για το

Εικόνα 2.20: Η εικόνα απεικονίζει ένα κινητό τηλέφωνο απλής χρήσης με το λειτουργικό KaiOS εγκατεστημένο.

KaiOS βασίζονται σε τεχνολογίες ιστού HTML, CSS και JavaScript και εκτελούνται από το Gecko. Η ανάπτυξή τους είναι αρκετά απλή, διότι οι παραπάνω τεχνολογίες ιστού είναι ευρέως διαδεδομένες και η εκμάθησή τους είναι σχετικά εύκολη. Υπάρχουν έτοιμα πρότυπα για την διευκόλυνση των νέων προγραμματιστών και οι πιο έμπειροι έχουν την δυνατότητα να δημιουργήσουν καινούρια πρότυπα. Υπάρχει πολύ μικρός ανταγωνισμός στην δημιουργία εφαρμογών για το KaiOS σε σχέση με τα προηγούμενα δύο λειτουργικά που αναφέρθηκαν παραπάνω, επειδή είναι πολύ μικρότερος ο αριθμός των συσκευών πληκτρολογίου σε σχέση με τα έξυπνα τηλέφωνα με οθόνη αφής και είναι διαφορετική η κατηγορία συσκευών που το τρέχουν. Στην εικόνα 2.20 παραπάνω φαίνεται το γραφικό περιβάλλον του KaiOS.

2.2 Η αρχιτεκτονική του Android

Το Android αποτελείται από έναν μεγάλο αριθμό διαφορετικών τμημάτων, με σκοπό την υποστήριξη των λειτουργιών των κινητών συσκευών. Αυτά χωρίζονται σε πέντε ενότητες και τέσσερα κυρίως στρωματά. Στο πιο χαμηλό επίπεδο βρίσκεται ο πυρήνας Linux ανοιχτού κώδικα ο οποίος αποτελείται από βιβλιοθήκες C/C++ οι οποίες αξιοποιούνται μέσω υπηρεσιών πλαισίου εφαρμογών (αγγλικά: application framework services). Για να μπορέσει μια εφαρμογή να έχει πρόσβαση σε αυτές τις βιβλιοθήκες χρειάζεται η εικονική μηχανή Dalvik JVM (Java Virtual Machine) όπου είναι και το πιο πάνω επίπεδο από τις βιβλιοθήκες. Ανεβαίνοντας ακόμα ένα επίπεδο συναντούμε το πλαίσιο εφαρμογών (application framework) και τέλος έχουμε τις κυρίες (αγγλικά: core) εφαρμογές του συστήματος, οι οποίες αποτελούνται από έναν email πελάτη, μιας εφαρμογής διαχείρισης SMS, ενός ημερολογίου, ενός browser, μιας εφαρμογής διαχείρισης επαφών, και άλλες οι οποίες είναι προ εγκατεστημένες στο λειτουργικό σύστημα Android (εικόνα 2.21). [15]



Εικόνα 2.21: Η εικόνα δείχνει επιγραμματικά τα επίπεδα του Android. [14] [15]

2.2.1 Πυρήνας του Linux

Μεταξύ όλων των στοιχείων, ο πυρήνας Linux παρέχει την κύρια λειτουργικότητα διεργασιών του λειτουργικού συστήματος στα έξυπνα κινητά και η Dalvik Virtual Machine (DVM) παρέχει την πλατφόρμα για την εκτέλεση των εφαρμογών Android. Είναι δηλαδή η καρδία όλου του συστήματος, διότι ως πυρήνας του Android περιέχει όλα τα βασικά προγράμματα οδήγησης υλικού όπως η κάμερα, το πληκτρολόγιο, η οθόνη και άλλα που χρειάζονται κατά την εκτέλεση των εφαρμογών. Με αυτόν τον τρόπο, διευκολύνει τη διασύνδεση με το περιφερειακό υλικό της συσκευής που τρέχει το λειτουργικό σύστημα. Επιπλέον παρέχει ένα επίπεδο διασύνδεσης μεταξύ του υλικού της συσκευής και των άλλων συστατικών στοιχείων του Android. Φυσικά έχουν γίνει αρκετές αλλαγές στον πυρήνα ώστε να είναι ελαφρύτερος για αυτού του είδους συσκευών και να βελτιστοποιείται η χρήση του σε κινητές συσκευές. [14] [15]

2.2.2 Βιβλιοθήκες πλατφόρμας του Android

Στο επόμενο επίπεδο έχουμε τις βιβλιοθήκες του Android οι οποίες είναι γραμμένες κυρίως σε γλώσσες προγραμματισμού C/C++ και Java οι οποίες είναι ενσωματωμένες στις εφαρμογές που αναπτύσσονται για το λειτουργικό σύστημα και παρέχουν υποστήριξη στην ανάπτυξη εφαρμογών για πολυμέσα, γραφικά, OpenGl και άλλα. Παρακάτω ακολουθούν μερικές βασικές βιβλιοθήκες με τις λειτουργίες τους:

- **SQLite** : Είναι η βιβλιοθήκη που παρέχει την υποστήριξη για βάσεις δεδομένων μικρού όγκου. Είναι μια δυναμική βάση δεδομένων η οποία χρησιμοποιείται από διάφορες εφαρμογές για την αποθήκευση των δεδομένων τους στην συσκευή.
- **FreeType** : Είναι μια πολύ ελαφριά βιβλιοθήκη η οποία χρησιμοποιείται για να δημιουργήσει τις γραμματοσειρές που παρουσιάζονται στο λειτουργικό.
- **Web-Kit** : Είναι μια μηχανή προγράμματος περιήγησης ανοιχτού κώδικα που παρέχει όλες τις λειτουργίες για την εμφάνιση περιεχομένου ιστού και απλοποιεί την φόρτωση των ιστοσελίδων.
- **Media** : Η βιβλιοθήκη αυτή παρέχει υποστήριξη για την αναπαραγωγή και εγγραφή διαφόρων μορφών ήχου και βίντεο.
- **SSL (Secure Sockets Layer):** Η τεχνολογία αυτή προσφέρει την εγκατάσταση ενός κρυπτογραφημένου συνδέσμου μεταξύ ενός διακομιστή ιστού και ενός προγράμματος περιήγησης ιστού. [14] [15]

2.2.3 Περιβάλλον χρόνου εκτέλεσης εφαρμογής

Το περιβάλλον χρόνου εκτέλεσης εφαρμογής (αγγλικά: application runtime) είναι το τρίτο επίπεδο στο λειτουργικό σύστημα Android το οποίο έχει μεγάλη σημασία επειδή περιέχει στοιχεία όπως τις βασικές βιβλιοθήκες και την εικονική μηχανή Dalvik (DVM). Κύριος σκοπός του είναι να παρέχει τη βάση για το πλαίσιο εφαρμογών και υποστήριξη των εφαρμογών με τη βοήθεια των βασικών βιβλιοθηκών. Η εικονική μηχανή Dalvik είναι ένα είδος εικονικής μηχανής Java ειδικά σχεδιασμένη και βελτιστοποιημένη για το Android και χρησιμοποιεί καταχωρητές για την λειτουργία της. Βασίζεται στο επίπεδο πυρήνα και χρησιμοποιεί χαρακτηριστικά όπως την διαχείριση μνήμης και την δυνατότητα να υπάρχουν πολυδιεργασίες (χαρακτηριστικό εγγενές με τη γλώσσα προγραμματισμού Java). Έτσι, η εικονική μηχανή Dalvik επιτρέπει σε κάθε εφαρμογή να τρέχει ξεχωριστά στη δική της διεργασία χωρίς καμιά επαφή με άλλη εφαρμογή έστω και αν εκτελούνται παράλληλα. Είναι

σχεδιασμένη με τέτοιο τρόπο έτσι ώστε να είναι εφικτό να τρέξουν πολλές εικονικές μηχανές ταυτόχρονα. [14] [15]

2.2.4 Πλαίσιο Εφαρμογής

Το Android προσφέρει στους προγραμματιστές μια πλατφόρμα ανάπτυξης ανοιχτού κώδικα δίνοντας τους έτσι τη δυνατότητα να αναπτύξουν με αυτή ιδιαιτέρα καινοτόμες και πλούσιες σε λειτουργίες εφαρμογές. Το επίπεδο πλαίσιο εφαρμογής (αγγλικά: application framework) παρέχει πολλές υπηρεσίες υψηλότερου επιπέδου σε εφαρμογές με τη μορφή κλάσεων Java. Η ανάπτυξη εφαρμογών γίνεται αρκετά πιο εύκολη καθώς υπάρχει η δυνατότητα ελέγχου του υλικού της συσκευής και μεσώ αυτής μπορούν να υποστηριχθούν λειτουργίες όπως υπηρεσίες εντοπισμού, εκτέλεση διεργασιών παρασκήνιου και πολλές ακόμη δυνατότητες. Τα σημαντικότερα δομικά στοιχειά του πλαισίου εφαρμογών είναι:

- **Διαχειριστής Δραστηριότητας**: Ελέγχει όλες τις πτυχές της ζωής των εφαρμογών και τη στοίβα δραστηριότητας.
- **Παροχείς περιεχομένου**: Επιτρέπει στις εφαρμογές τη δημοσίευση και την κοινή χρήση δεδομένων με άλλες εφαρμογές.
- **Διαχειριστής πόρων**: Παρέχει πρόσβαση σε μη-κώδικα ενσωματωμένων πόρων, όπως συμβολοσειρές, ρυθμίσεις χρωμάτων και διατάξεις διεπαφής χρήστη.
- **Διαχειριστής ειδοποιήσεων**: Επιτρέπει στις εφαρμογές την εμφάνιση ειδοποιήσεων στο χρήστη.
- **Όψη συστήματος**: Είναι ένα επεκτάσιμο σύνολο των όψεων που χρησιμοποιείται για τη δημιουργία διεπαφών χρήστη της εφαρμογής. [14] [15]

2.2.5 Εφαρμογές

«Οι εφαρμογές είναι το ανώτερο επίπεδο της αρχιτεκτονικής Android. Οι προ εγκατεστημένες εφαρμογές όπως η αρχική σελίδα, οι επαφές, η κάμερα, η συλλογή κ.λπ. και οι εφαρμογές τρίτων που έχουν ληφθεί από το playstore, θα εγκατασταθούν μόνο σε αυτό το επίπεδο. Τρέχει εντός του χρόνου εκτέλεσης του Android με τη βοήθεια των κλάσεων και των υπηρεσιών που παρέχονται από το πλαίσιο εφαρμογής.» [15]

2.3 Βασικές αρχές εφαρμογών Android

Οι εφαρμογές για το λειτουργικό Android μπορούν να αναπτυχθούν χρησιμοποιώντας τις γλώσσες αντικειμενοστραφή προγραμματισμού Kotlin, Java και C ++. Για την δημιουργία των εφαρμογών χρησιμοποιούμε τα εργαλεία που παρέχει η Google τα οποία λέγονται Android SDK ή Android Software Development Kit (ελληνικά: κιτ ανάπτυξης λογισμικού Android). Το Android SDK είναι ένα σύνολο εργαλείων, οδηγιών και προγραμμάτων που χρησιμοποιούνται για την ανάπτυξη εφαρμογών στην πλατφόρμα του Android. Το SDK περιλαμβάνει API ή application programming interface (ελληνικά: διεπαφή προγραμματισμού εφαρμογών), IDE ή integrated development environment (ελληνικά: ολοκληρωμένο περιβάλλον ανάπτυξης), τεκμηρίωση, βιβλιοθήκες, δείγματα κώδικα και άλλα βοηθητικά προγράμματα. Τα εργαλεία του Android SDK μεταγλωττίζουν τον πηγαίο κώδικά μαζί με τυχόν αρχεία δεδομένων και πόρων σε ένα APK ή android package kit, ένα πακέτο Android, το οποίο είναι ένα αρχείο με επίθημα .apk. Ένα αρχείο APK περιέχει όλα τα περιεχόμενα μιας εφαρμογής Android και είναι το αρχείο που χρησιμοποιούν συσκευές Android για την εγκατάσταση της εφαρμογής.

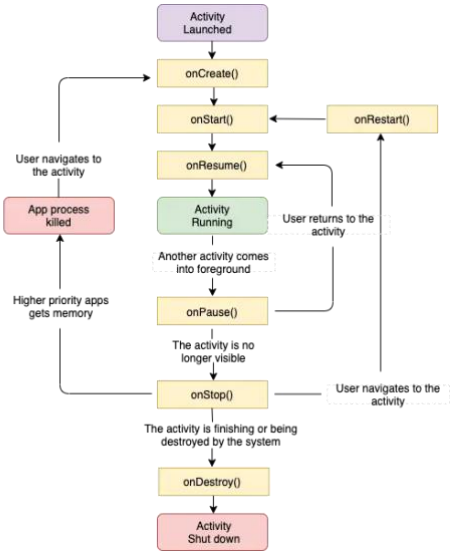
Κάθε εφαρμογή έχει το δικό της περιβάλλον ασφαλείας (sandbox ασφάλεια) και προστατεύεται από κάποιες λειτουργίες ασφαλείας που υποστηρίζει το Android.

- Το λειτουργικό σύστημα Android είναι ένα σύστημα Linux πολλαπλών χρηστών στο οποίο κάθε εφαρμογή αναπαριστά ένα διαφορετικό χρήστη.
- Από προεπιλογή, το σύστημα εκχωρεί σε κάθε εφαρμογή ένα μοναδικό αναγνωριστικό χρήστη Linux (το αναγνωριστικό χρησιμοποιείται μόνο από το σύστημα και δεν είναι προσβάσιμο από την εφαρμογή). Το σύστημα ορίζει δικαιώματα για όλα τα αρχεία σε μια εφαρμογή, έτσι ώστε μόνο το αναγνωριστικό χρήστη που έχει εκχωρηθεί σε αυτήν την εφαρμογή να έχει πρόσβαση σε αυτά.
- Κάθε διαδικασία έχει τη δική της εικονική μηχανή (VM), οπότε ο κώδικας μιας εφαρμογής εκτελείται ξεχωριστά από άλλες εφαρμογές.
- Από προεπιλογή, κάθε εφαρμογή εκτελείται με τη δική της διαδικασία Linux. Το σύστημα Android ξεκινά τη διαδικασία όταν οποιοδήποτε από τα στοιχεία της εφαρμογής πρέπει να εκτελεστεί και, στη συνέχεια, τερματίζει τη διαδικασία όταν δεν χρειάζεται πλέον ή όταν το σύστημα πρέπει να ανακτήσει μνήμη για άλλες εφαρμογές. [16]

Το σύστημα Android έχει σαν αρχή να δίνει τα λιγότερα προνομία. Δηλαδή κάθε εφαρμογή, από προεπιλογή, έχει πρόσβαση μόνο στα στοιχεία που απαιτεί για να εκτελέσει τη διεργασία της. Αυτό δημιουργεί ένα πολύ ασφαλές περιβάλλον στο οποίο μια εφαρμογή δεν μπορεί να αποκτήσει πρόσβαση σε τμήματα του συστήματος για τα οποία δεν έχει άδεια. Ωστόσο υπάρχουν τρόποι για μια εφαρμογή να μοιράζεται δεδομένα με άλλες εφαρμογές και για μια εφαρμογή να έχει πρόσβαση στις υπηρεσίες συστήματος. [16]

2.3.1 Δομικά στοιχεία των εφαρμογών

Τα στοιχεία που παρουσιάζονται παρακάτω είναι τα απαραίτητα δομικά στοιχεία μιας εφαρμογής Android. Αυτά τα στοιχεία συνδυάζονται με το αρχείο δήλωσης εφαρμογής (αρχείο AndroidManifest) που περιγράφει κάθε στοιχείο της εφαρμογής και πώς αλληλοεπιδρούν. Κάθε στοιχείο είναι ένα σημείο εισόδου μέσω του οποίου το σύστημα ή ένας χρήστης μπορεί να εισέλθει στην εφαρμογή. Ορισμένα μέρη εξαρτώνται από άλλα. Υπάρχουν τέσσερις διαφορετικοί κύριοι τύποι στοιχείων εφαρμογής: δραστηριότητες (activities), υπηρεσίες (services), δέκτες μετάδοσης (broadcast receivers) και πάροχος περιεχομένου (content provider).



Εικόνα 2.22: Κύκλος ζωής της δραστηριότητας (activity).

Κάθε τύπος εξυπηρετεί έναν ξεχωριστό σκοπό και έχει έναν ξεχωριστό κύκλο ζωής που καθορίζει τον τρόπο δημιουργίας και καταστροφής της δραστηριότητας (activity). Το παραπάνω σχήμα (εικόνα 2.22) δείχνει τον κύκλο ζωής που έχει μια δραστηριότητα (activity). [16]

2.3.2 Δραστηριότητες

Μια δραστηριότητα (αγγλικά: activity) είναι το «σημείο εισόδου» για την αλληλεπίδραση τους εφαρμογής με τον χρήστη. Αντιπροσωπεύει μία οθόνη με διεπαφή χρήστη. Για παράδειγμα, μια εφαρμογή ηλεκτρονικού ταχυδρομείου μπορεί να έχει μια δραστηριότητα που εμφανίζει μια λίστα με νέα μηνύματα ηλεκτρονικού ταχυδρομείου, μια άλλη δραστηριότητα για τη σύνταξη τους μηνύματος ηλεκτρονικού ταχυδρομείου και μια άλλη δραστηριότητα για την ανάγνωση μηνυμάτων ηλεκτρονικού ταχυδρομείου. Παρόλο που οι δραστηριότητες συνεργάζονται για να δημιουργήσουν μια συνολική εμπειρία χρήστη στην εφαρμογή email, κάθε μία είναι ανεξάρτητη από τις άλλες. [16]

2.3.3 Υπηρεσίες

Μια υπηρεσία (αγγλικά: service) είναι ένα στοιχείο που εκτελείται στο παρασκήνιο για να εκτελεί μακροχρόνιες λειτουργίες. Για παράδειγμα, μια υπηρεσία μπορεί να αναπαράγει μουσική στο παρασκήνιο ενώ ο χρήστης βρίσκεται σε διαφορετική εφαρμογή ή μπορεί να ανακτήσει δεδομένα μέσω του δικτύου χωρίς να εμποδίσει την αλληλεπίδραση του χρήστη με μια δραστηριότητα. Οι υπηρεσίες δεν παρέχουν γραφικό περιβάλλον. Υπάρχουν στην ουσία δύο διαφορετικές σημασιολογικές υπηρεσίες που λένε στο σύστημα πώς να διαχειρίζεται μια εφαρμογή. Οι υπηρεσίες που ξεκίνησαν λένε στο σύστημα να τις διατηρήσει σε λειτουργία μέχρι να ολοκληρωθεί η εργασία τους. Οι δεσμευμένες υπηρεσίες εκτελούνται επειδή κάποια άλλη εφαρμογή (ή το σύστημα) δήλωσε ότι θέλει να κάνει χρήση της υπηρεσίας. [16]

2.3.4 Δέκτες εκπομπής

Οι δέκτες εκπομπής (αγγλικά: broadcast receivers) απαντούν σε μηνύματα μετάδοσης από τις εφαρμογές ή από το σύστημα. Για παράδειγμα, οι εφαρμογές μπορούν να ξεκινήσουν εκπομπές για να ενημερώσουν τις εφαρμογές ότι ορισμένα δεδομένα έχουν ληφθεί στη συσκευή και είναι διαθέσιμα για χρήση, οπότε ο δέκτης εκπομπής είναι αυτός που θα παρακολουθεί αυτήν την επικοινωνία και θα ξεκινήσει την κατάλληλη ενέργεια. [16]

2.3.5 Πάροχοι περιεχομένου

Ο πάροχος περιεχομένου παρέχει δεδομένα από μία εφαρμογή σε άλλη, κατόπιν αιτήματος. Διαχειρίζεται την πρόσβαση σε ένα δομημένο σύνολο δεδομένων. Οι πάροχοι περιεχομένου είναι η τυπική διεπαφή που συνδέει τα δεδομένα σε μία διαδικασία με τον κώδικα που εκτελείται σε μια άλλη διαδικασία. Τα δεδομένα μπορούν να αποθηκευτούν στο σύστημα αρχείων τους συσκευής Android, σε βάση δεδομένων ή κάπου αλλού. Το ίδιο το Android περιλαμβάνει παρόχους περιεχομένου που διαχειρίζονται δεδομένα ήχου, βίντεο, εικόνων και προσωπικά στοιχεία επικοινωνίας. Σε περιπτώσεις όπου χρειάζεται να μεταφερθούν πολύπλοκα δεδομένα ή αρχεία, απαιτείται ο ορισμός νέου παρόχου από τον προγραμματιστή. Με ορισμένους περιορισμούς, οι πάροχοι έχουν πρόσβαση σε οποιαδήποτε εφαρμογή Android. [16]

2.3.6 Πρόσθετα στοιχεία του Android

Τρία από τα βασικά στοιχεία μιας εφαρμογής, οι δραστηριότητες, οι υπηρεσίες και οι δέκτες εκπομπής, ενεργοποιούνται μέσω μηνυμάτων που ονομάζονται προθέσεις (αγγλικά: intents). Η πρόθεση ανταλλαγής μηνυμάτων είναι μια δυνατότητα δέσμευσης χρόνου εκτέλεσης μεταξύ στοιχείων στις εφαρμογές. Είναι στην ουσία μια αφηρημένη περιγραφή της λειτουργίας που πρέπει να εκτελεστεί. Υπάρχουν ξεχωριστοί μηχανισμοί για την παροχή προθέσεων σε κάθε τύπο «συστατικού». Σε κάθε περίπτωση, το σύστημα βρίσκει την κατάλληλη δραστηριότητα, την υπηρεσία ή το σύνολο των δεκτών εκπομπής για να ανταποκριθεί στην πρόθεση. Δεν υπάρχει αλληλοεπικάλυψη σε αυτά τα συστήματα ανταλλαγής μηνυμάτων. Παραδείγματος χάριν, οι προθέσεις εκπομπής παρέχονται μόνο σε δέκτες εκπομπής, ποτέ σε δραστηριότητες ή υπηρεσίες. [14]

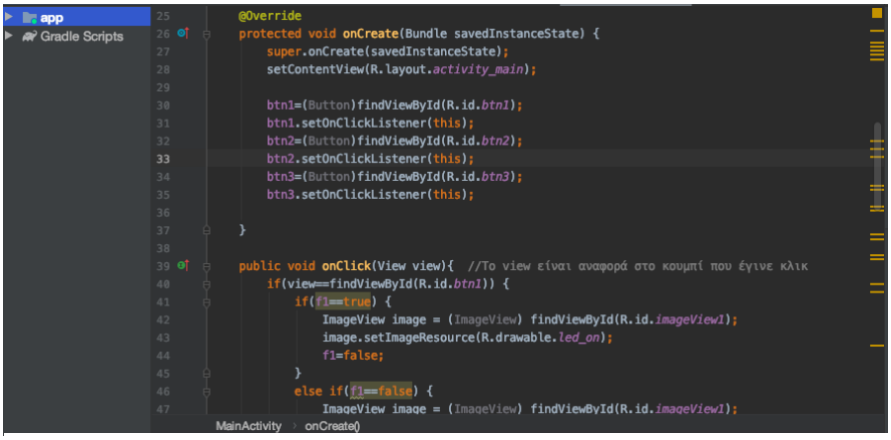
Κεφάλαιο 3: Εργαλεία ανάπτυξης μιας εφαρμογής Android

Σε αυτό το κεφάλαιο παρουσιάζονται επιγραμματικά τα εργαλεία που απαιτούνται για την ανάπτυξη τους εφαρμογής που παρουσιάζεται σε αυτή την εργασία, τα οποία είναι ευρέως διαδεδομένα για τέτοιου είδους ανάπτυξης λογισμικού.

Εικόνα 3.1 Λογότυπο του Android SDK

3.1 Android SDK

Το Android SDK, που προαναφέρθηκε στην ενότητα 2.3, είναι μια συλλογή από εργαλεία ανάπτυξης λογισμικού και βιβλιοθήκες που απαιτούνται για την ανάπτυξη εφαρμογών Android. Με κάθε νέα έκδοση Android ή μια ενημέρωση που η Google κυκλοφορεί, διαθέτει ένα αντίστοιχο SDK, το οποίο πρέπει να κατεβάσουν και να εγκαταστήσουν οι προγραμματιστές. Αξίζει να σημειωθεί ότι μπορεί να γίνει λήψη και χρήση του Android SDK ανεξάρτητα από το Android Studio, αλλά συνηθίζεται να αξιοποιείται μέσω του Android Studio για οποιαδήποτε ανάπτυξη εφαρμογής Android. Περιλαμβάνει όλα τα απαραίτητα εργαλεία για τον κώδικα προγραμμάτων από το μηδέν ακόμη και για τη δοκιμή των εφαρμογών. Αυτά τα εργαλεία καθιστούν ομαλή όλη την ροή της διαδικασίας από την ανάπτυξη και τον εντοπισμό σφαλμάτων, έως την τελική εφαρμογή. Το Android SDK είναι συμβατό με όλα τα δημοφιλή λειτουργικά, πιο συγκεκριμένα τα Windows, macOS και Linux, ώστε να είναι δυνατό να γίνει ανάπτυξη σε οποιαδήποτε από αυτές τις πλατφόρμες.



Εικόνα 3.2 Κώδικας σε γλώσσα προγραμματισμού Java σε περιβάλλον Android Studio.

3.1.1 Γλώσσες προγραμματισμού

Η Google δηλώνει ότι «Οι εφαρμογές Android μπορούν να γραφτούν χρησιμοποιώντας γλώσσες Kotlin, Java και C ++», χρησιμοποιώντας το κιτ ανάπτυξης λογισμικού Android (Android SDK), ενώ είναι δυνατή η χρήση και άλλων γλωσσών προγραμματισμού. Ο ενδεδειγμένος και ευκολότερος τρόπος ανάπτυξης εφαρμογών Android είναι η χρήση της

γλώσσας προγραμματισμού Java (εικόνα 3.2). Παρόλο που η Java είναι ένα εργαλείο γενικής χρήσης, χρησιμοποιείται σε συνδυασμό με το Android SDK σε περιβάλλον Android Studio για την δημιουργία εφαρμογών. Οι γλώσσες που δεν χρησιμοποιούν την εικονική μηχανή της Java (JVM), όπως η JavaScript, C, C ++ και η assembly, χρειάζονται τη βοήθεια του κώδικα γλώσσας JVM η οποία μπορεί να παρέχεται σε κάποιες περιπτώσεις από εργαλεία, πιθανώς με περιορισμένη υποστήριξη API. Ο επίσημος τρόπος ανάπτυξης είναι να χρησιμοποιηθεί η C ++ με το Native Development Kit (NDK). Με αυτό τον τρόπο γίνεται ανάπτυξη εφαρμογών με εντολές χαμηλού επιπέδου, με χρονικά ευαίσθητα προγράμματα οδήγησης. Χρησιμοποιώντας την C ++ και το NDK, μπορούμε να εκτελέσουμε απευθείας την εφαρμογή στον πυρήνα Android, αυξάνοντας έτσι την αποδοτικότητα σε αντάλλαγμα του μήκους κώδικα και του κόστους ανάπτυξης. Υπάρχουν όμως εργαλεία τρίτων όπως το Xamarin, το Crodova και το React Native για την ανάπτυξη εφαρμογών. [16]

3.1.2 Χαρακτηριστικά του Android SDK

Το κιτ ανάπτυξης λογισμικού Android περιλαμβάνει ένα ολοκληρωμένο σύνολο εργαλείων. Σε αυτά περιλαμβάνονται το πρόγραμμα εντοπισμού σφαλμάτων, οι βιβλιοθήκες, ο εξομοιωτής συσκευών που βασίζεται σε QEMU, η τεκμηρίωση, το δείγμα κώδικα και τα σεμινάρια. Μέχρι το τέλος του έτους 2014 το επίσημα υποστηριζόμενο περιβάλλον ανάπτυξης (IDE) ήταν το Eclipse κάνοντας χρήση των εργαλείων ανάπτυξης Android (ADT). Παρόλο που οι προγραμματιστές έχουν την ελευθερία να επιλέξουν το περιβάλλον ανάπτυξης της προτίμησης τους από το έτος 2015 και μετά το ADT καταργήθηκε και η Google εστίασε στο περιβάλλον Android Studio. Το Android SDK είναι βελτιστοποιημένο για το Android Studio και επομένως για να αποκομίσουν αποτελεσματικά τα οφέλη του οι προγραμματιστές, θα πρέπει να το εγκαταστήσουν. Η διαχείριση του SDK Android από το Android Studio είναι ευκολότερη, καθώς η υποστήριξη για γλώσσες τους Java, Kotlin και C ++ γίνεται αυτόματα. Εκτός από αυτό, η εγκατάσταση των ενημερώσεων στο Android SDK εκτελείται αυτόματα από το Android Studio. Παρακάτω γίνεται μια σύντομη αναφορά στα κύρια χαρακτηριστικά που υποστηρίζει:

- Βελτιστοποιημένη Dalvik εικονική μηχανή για κινητές συσκευές.
- Εφαρμογή πλαισίου που επιτρέπει την επαναχρησιμοποίηση και την αντικατάσταση στοιχειών.
- SQLite για αποθήκευση δεδομένων.
- GSM Τηλεφωνία.
- Υποστήριξη πολυμέσων για αρχεία ήχου, βίντεο και εικόνων (MPEG4, H.264, MP3, AAC, AMR, JPG, PNG, GIF).
- Εργαλεία για τον εντοπισμό σφαλμάτων.
- Ολοκληρωμένο πρόγραμμα περιήγησης.
- GPS και πυξίδα.

3.1.3 Εργαλεία πλατφόρμας του Android SDK

Τα εργαλεία πλατφόρμας (αγγλικά: Android SDK Platform-Tools) είναι ένα στοιχείο του Android SDK. Περιλαμβάνει εργαλεία που αλληλοεπιδρούν με την πλατφόρμα Android, τα adb, fastboot και systrace. Αυτά τα εργαλεία απαιτούνται για την ανάπτυξη εφαρμογών Android και χρησιμοποιούνται για την υποστήριξη των λειτουργιών για την τρέχουσα έκδοση της πλατφόρμας Android. Αυτά τα εργαλεία διασυνδέονται με την πλατφόρμα Android στη

συσκευή που χρησιμοποιούν οι προγραμματιστές για την δοκιμή των εφαρμογών. Το Android SDK περιλαμβάνει:

- *Android Debug Bridge (adb)*: Αυτό είναι ένα εύχρηστο εργαλείο γραμμής εντολών που επιτρέπει την επικοινωνία με μια συσκευή. Η εντολή adb επιτρέπει την εκτέλεση ενεργειών στην συσκευή, όπως η εγκατάσταση και ο εντοπισμός σφαλμάτων εφαρμογών. Παρέχει πρόσβαση σε ένα κέλυφος Unix που μπορεί να χρησιμοποιηθεί για να εκτελεστεί μια ποικιλία εντολών σε μια συσκευή.
- *Fastboot*: Είναι ένα πρωτόκολλο και διαθέτει ένα εργαλείο με το ίδιο όνομα που περιλαμβάνεται στο πακέτο Android SDK. Χρησιμοποιείται κυρίως για την τροποποίηση του συστήματος αρχείων flash (για συσκευές που βασίζονται στην μνήμη τύπου flash) μέσω σύνδεσης USB από κεντρικό υπολογιστή. Το εργαλείο αυτό επιτρέπει την εγκατάσταση σε μια συσκευή με μια νέα «εικόνα» συστήματος.
- *Android Asset Packaging Tool (AAPT)*: Το εργαλείο αυτό δίνει την δυνατότητα δημιουργίας .apk αρχείων, τα οποία περιέχουν τα εκτελέσιμα αρχεία και τους πόρους μιας εφαρμογής.
- *Systrace*: Αυτό το εργαλείο βοηθά στη συλλογή και τον έλεγχο πληροφοριών χρονισμού σε διαδικασίες που εκτελούνται στη συσκευή Android σε επίπεδο συστήματος. Είναι σημαντικό για τον εντοπισμό σφαλμάτων στην απόδοση της εφαρμογής.
- *Βιβλιοθήκη sqlite3*: Η βιβλιοθήκη αυτή επιτρέπει την πρόσβαση στα δεδομένα της SQLite βάσης δεδομένων που δημιουργούνται από τις διάφορες εφαρμογές τοπικά στην συσκευή.
- *Android Emulator*: Είναι ένας εξομοιωτής συσκευών (εικόνα 2.3) βασισμένος σε QEMU που προσομοιώνει συσκευές Android στον υπολογιστή, επιτρέποντας στους προγραμματιστές να δοκιμάζουν εφαρμογές σε διαφορετικές συσκευές και επίπεδα API Android, χωρίς να χρειάζεται να έχουν πραγματικές συσκευές για κάθε μία έκδοση. Ο εξομοιωτής διαθέτει διαμορφώσεις για διάφορα Android τηλέφωνα, tablet, Wear OS ρολόγια και Android TV συσκευές. Ο εξομοιωτής παρέχει ένα μεγάλο πλήθος λειτουργιών μιας τυπικής συσκευής, η οποία τρέχει Android, δηλαδή σχεδόν όλες οι λειτουργίες που γίνονται σε μια τέτοια συσκευή μπορούν να χρησιμοποιηθούν. Για παράδειγμα μπορούν να γίνουν τηλεφωνικές κλήσεις και να σταλθούν μηνύματα, να ληφθούν δεδομένα θέσης από τους χάρτες, υπάρχει δυνατότητα πρόσβασης στο διαδίκτυο και στο Google Play Store, παρέχονται λειτουργίες για την εύκολή αποσφαλμάτωση και πολλά άλλα.
- *Android NDK*: Ο κώδικας που γράφεται σε γλώσσες C / C ++ μπορεί να μεταγλωττιστεί σε επεξεργαστή ARM ή σε εγγενή κώδικα x86 χρησιμοποιώντας το Native Development Android Kit (NDK) που προαναφέρθηκε στο κεφάλαιο 2.1.1. Το NDK χρησιμοποιεί τον μεταγλωττιστή Clang για να μεταγλωττίσει σε C / C ++.

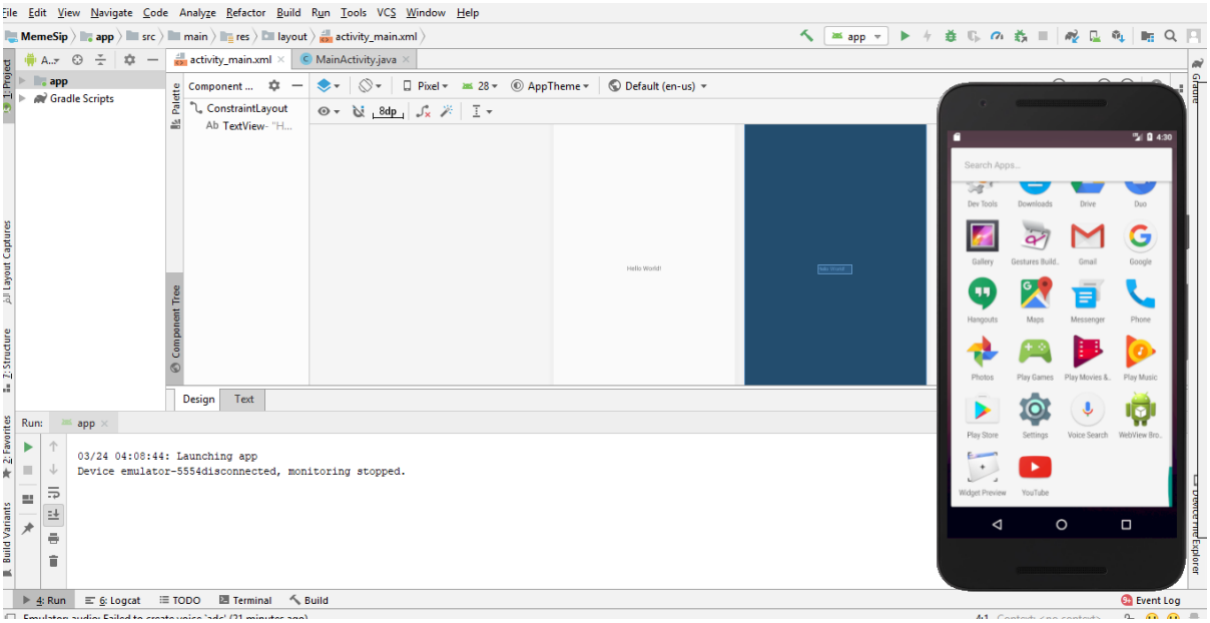
3.2 Android Studio

Το Android Studio είναι ένα ολοκληρωμένο προγραμματιστικό περιβάλλον (IDE) για ανάπτυξη εφαρμογών στην πλατφόρμα Android, βασισμένο στο IntelliJ IDEA. Το Android Studio παρέχει ένα ενοποιημένο περιβάλλον όπου μπορούν να δημιουργηθούν εφαρμογές για τηλέφωνα Android, tablet, Android Wear ρολόγια, Android τηλεοράσεις και Android Auto συστήματα (εικόνα 3.3). Οι δομικές μονάδες κώδικα επιτρέπουν την διαίρεση του έργου ενός προγραμματιστή σε μονάδες λειτουργικότητας που μπορούν να δημιουργηθούν, να δοκιμαστούν και να γίνει εντοπισμός σφαλμάτων ανεξάρτητα. Όπως αναφέρθηκε

προηγουμένως, η επίσημη γλώσσα για την ανάπτυξη Android είναι η Java. Μεγάλα τμήματα του Android είναι γραμμένα στην Java και τα API του έχουν σχεδιαστεί για να καλούνται κυρίως από την Java. Είναι δυνατή η ανάπτυξη εφαρμογών C και C ++ χρησιμοποιώντας το Android Native Development Kit (NDK), ωστόσο δεν είναι κάτι που προωθεί η Google. Βασισμένο στο λογισμικό της JetBrains (το IntelliJ IDEA), το Android Studio σχεδιάστηκε αποκλειστικά για προγραμματισμό Android. Είναι διαθέσιμο για Windows, Mac OS X και Linux και αντικατέστησε το Eclipse Android Development Tools (ADT) ως το κύριο IDE της Google για ανάπτυξη εφαρμογών Android. Η πρώτη σταθερή έκδοση βγήκε το Δεκέμβριο του 2014, με την έκδοση 1.0 (οι εκδόσεις 0.1 και 0.8 ήταν δοκιμαστικές).

Το περιβάλλον αυτό ενσωματώνει το Android SDK και προσφέρει ακόμη περισσότερες δυνατότητες που βελτιώνουν την παραγωγικότητά κατά τη δημιουργία εφαρμογών Android [17], όπως:

- Ένα ευέλικτο σύστημα κατασκευής με βάση το Gradle.
- Έναν γρήγορο εξομοιωτή πλούσιο σε χαρακτηριστικά.
- Ένα ενοποιημένο περιβάλλον όπου είναι δυνατή η ανάπτυξη για όλες τις συσκευές Android.
- Εφαρμογή αλλαγών για την προώθηση των αλλαγών στον κώδικα και στους πόρους στην τρέχουσα εφαρμογή που δημιουργείται χωρίς επανεκκίνηση της εφαρμογής.
- Πρότυπα κώδικα και ενσωμάτωση στο GitHub για βοηθήσουν στην δημιουργία κοινών δυνατοτήτων εφαρμογών και για την εισαγωγή δείγμα κώδικα.
- Εκτεταμένα εργαλεία δοκιμής και πλαίσια.
- Εργαλεία Lint για τον εντοπισμό της απόδοσης, της χρηστικότητας, της συμβατότητας έκδοσης και άλλων προβλημάτων.



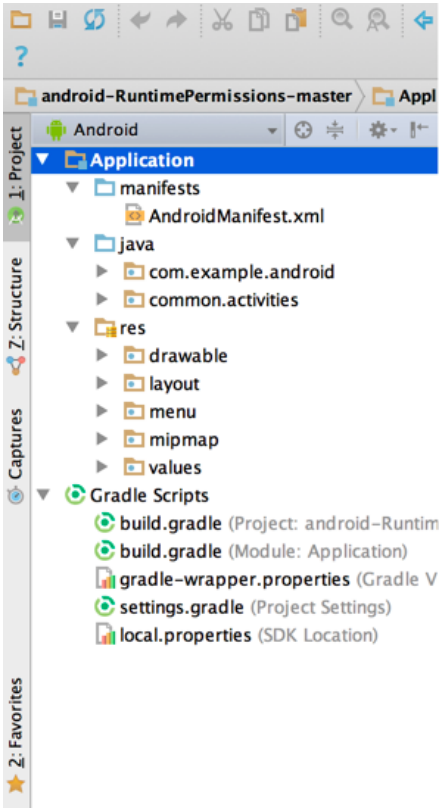
Εικόνα 3.3 Αποτύπωμα οθόνης από το πρόγραμμα Android Studio με ανοιχτό τον εξομοιωτή.

- Ενσωματωμένη υποστήριξη για το Google Cloud Platform, διευκολύνοντας την ενσωμάτωση του Google Cloud Messaging και του App Engine.

Δομή έργων

Κάθε έργο στο Android Studio περιέχει μία ή περισσότερες λειτουργικές μονάδες με αρχεία πηγαίου κώδικα και αρχεία πόρων. Οι τύποι των ενοτήτων περιλαμβάνουν τις ενότητες εφαρμογών Android, τις ενότητες βιβλιοθήκης και τις ενότητες Google App Engine. Από προεπιλογή, το Android Studio εμφανίζει τα αρχεία έργου στην προβολή έργου Android, όπως φαίνεται στην εικόνα 3.4. Αυτή η προβολή οργανώνεται από λειτουργικές μονάδες για γρήγορη πρόσβαση στα βασικά αρχεία προέλευσης του έργου. [17]

Όλα τα αρχεία build είναι ορατά στο ανώτερο επίπεδο στο Gradle Scripts και κάθε



Εικόνα 3.4 Τα αρχεία του έργου στο android view.

ενότητα εφαρμογής περιέχει τους φακέλους manifests (περιέχει το αρχείο AndroidManifest.xml), java (περιέχει τα αρχεία πηγαίου κώδικα Java, συμπεριλαμβανομένου του δοκιμαστικού κώδικα JUnit) και res (περιέχει όλους τους πόρους χωρίς κώδικα, όπως διατάξεις XML, συμβολοσειρές UI και εικόνες bitmap). [17]

3.3 phpMyAdmin

Το phpMyAdmin είναι ένα ανοιχτού κώδικα εργαλείο λογισμικού γραμμένο σε PHP, το οποίο προορίζεται για τη διαχείριση της MySQL βάσης δεδομένων μέσω του Διαδικτύου. Έχει γίνει ένα από τα πιο δημοφιλή εργαλεία διαχείρισης MySQL, ειδικά για υπηρεσίες φιλοξενίας ιστοσελίδων αλλά και εφαρμογές. Για την κατανόηση του εργαλείου αυτού γίνεται παρακάτω μια σύντομη παρουσίαση γενικών εννοιών για τις βάσεις δεδομένων, την MySQL και την SQLite.

3.3.1 Βασικές έννοιες βάσεων δεδομένων

Ως βάση δεδομένων ορίζουμε ένα σύνολο πινάκων που περιέχουν δεδομένα και συνδέονται μεταξύ τους με λογικές σχέσεις (σχεσιακή βάση δεδομένων). Οι βασικές απαιτήσεις που πρέπει να ικανοποιεί μία βάση δεδομένων για να είναι σωστή και ολοκληρωμένη είναι οι ακόλουθες:

- Να παρέχεται γρήγορη πρόσβαση στις πληροφορίες που απαιτούνται από κάθε χρήστη. Η ταχύτητα έχει μεγάλη σημασία για την αποδοτικότητα του συστήματος βάσης δεδομένων.
- Να εκπληρώνει τις ανάγκες για δεδομένα με ένα συνεπή και οικονομικό τρόπο.
- Να απαλείψει ή και να ελαχιστοποιήσει διπλά δεδομένα των περιεχομένων των βάσεων δεδομένων ώστε να υπάρχουν μόνο τα απαραίτητα και να γίνεται σωστή αξιοποίηση του χώρου που είναι διαθέσιμο από το υλικό του συστήματος.
- Η βάση δεδομένων να είναι επεκτάσιμη και προσαρμόσιμη ανάλογα με τις απαιτήσεις που προκύπτουν με την πάροδο του χρόνου και την πιθανή εξέλιξη που θα επέλθει.

Για να καταστεί δυνατή η κατανόηση της έννοιας «βάση δεδομένων» είναι ωφέλιμο να αναφερθούν οι παρακάτω ορισμοί. Ως βασική πηγή πληροφορίων που αντιπροσωπεύονται σε μία βάση δεδομένων έχουμε τα αντικείμενα του πραγματικού κόσμου τα οποία ονομάζουμε οντότητες και για την άντληση τους σε πολλές περιπτώσεις είναι αναγκαίο να γίνει εκτεταμένη έρευνα σε δεδομένα και περιπτώσεις που αφορούν την πραγματική ζωή.

Οντότητα: Μία μοναδική αναπαράσταση ενός μοναδικού αντικειμένου του πραγματικού κόσμου, που δημιουργείται χρησιμοποιώντας τις τιμές των ιδιοτήτων σε μορφή αναγνώσιμη από υπολογιστή. Ιδιότητα: Μία σημαντική ιδιότητα ενός πραγματικού αντικειμένου. Κάθε ιδιότητα έχει μία τιμή που βοηθά στον προσδιορισμό της οντότητας για να την ξεχωρίσει από τα άλλα μέρη. Οι ιδιότητες περιέχονται σε πεδία πινάκων ή σε στήλες ερωτημάτων. Τύπος δεδομένων ιδιότητας: Οι βασικοί τύποι αποτελούνται από όλους τους αριθμητικούς τύπους δεδομένων (ακέραιους, πραγματικούς απλής ακρίβειας, διπλής ακρίβειας κ.τ.λ.) και συμβολοσειρές (κείμενο ή αλφαριθμητικό) χωρίς ενσωματωμένα κενά ή στίξη. Οι συμβολοσειρές μπορούν να περιέχουν γράμματα, αριθμούς ή ειδικούς χαρακτήρες. Πίνακας: Η συλλογή όλων των οντοτήτων ενός τύπου δεδομένων. Βάση Δεδομένων: Το σύνολο από πίνακες που αποθηκεύουν σχετιζόμενες οντότητες.

Μοντελοποίηση δεδομένων

Το πρωταρχικό βήμα στη σχεδίαση μίας βάσης δεδομένων είναι ο προσδιορισμός των οντοτήτων που αντιπροσωπεύονται μέσα στη βάση και ποιες από τις ιδιότητές τους θα περιλαμβάνονται. Αυτή η διαδικασία ονομάζεται μοντελοποίηση δεδομένων. Είναι πολύ σημαντική για την σχεδίαση της βάσης διότι έτσι ο σχεδιασμός θα είναι ορθός από την αρχή και δεν θα αναγκάσει τον προγραμματιστή μιας εφαρμογής ή μιας ιστοσελίδας να αναθεωρήσει την δομή του συστήματος κάνοντας πολλές μεγάλες αλλαγές. Ο σκοπός ενός μοντέλου δεδομένων είναι να οριστεί μία λογική αναπαράσταση της δομής των δεδομένων που χρησιμοποιούνται για να δημιουργηθεί μία βάση δεδομένων. Το μοντέλο δεδομένων μπορεί να περικλείει μία ολόκληρη εταιρεία, ένα τμήμα ή έναν τύπο αντικειμένων. Τα μοντέλα που σχετίζονται με αντικείμενα ονομάζονται λογικά μοντέλα δεδομένων. Υπάρχουν πολλές μέθοδοι για τη δημιουργία διαγραμμάτων εμφάνισης μοντέλου δεδομένων. Μία από τις πιο χρήσιμες μεθόδους που χρησιμοποιούνται είναι το διάγραμμα οντότητας – σχέσης (Entity-Relationship: E- R) που αναπτύχθηκε από τον Peter Chen το 1976. Ένα από τα βασικά πλεονεκτήματα του διαγράμματος E-R είναι ότι μπορούμε να το χρησιμοποιήσουμε για να

εμφανίσουμε τη θεωρητική σχεδίαση πολύ μεγάλων συστημάτων με πολλές βάσεις δεδομένων, σε σχετικά μικρό χώρο.

3.3.2 MySQL

Η MySQL, αποτελεί ένα από τα πιο δημοφιλή συστήματα διαχείρισης σχεσιακών βάσεων δεδομένων Open Source SQL, το οποίο αναπτύσσεται, διανέμεται και υποστηρίζεται από την Oracle Corporation. Μια βάση δεδομένων είναι ουσιαστικά μία συλλογή δεδομένων κατάλληλα διαμορφωμένων σε έναν υπολογιστή προκειμένου να μπορούν να προσπελαστούν και να ενημερωθούν με ευκολία από το άτομο που τις διαχειρίζεται. Οι ενέργειες αυτές επάνω

Εικόνα 3.5 MySQL λογότυπο.

στα δεδομένα γίνονται μέσα από τα συστήματα διαχείρισης βάσεων δεδομένων (DataBase Management Systems - DBMS), τα οποία έχουν κατάλληλους μηχανισμούς για να προσπελαύνουν οργανωμένες δομές δεδομένων και να τις επεξεργάζονται χωρίς απώλειες και κίνδυνο παραποίησης ή φθοράς των δεδομένων.

Η MySQL είναι στην ουσία μία γλώσσα προγραμματισμού για συστήματα DBMS. Προκειμένου τα DBMS να μπορούν να έχουν κάποιους κοινούς άξονες αναφοράς έχει αναπτυχθεί η γλώσσα SQL (Structured Query Language) η οποία επιτρέπει την προσπέλαση των δομημένων δεδομένων που περιέχει ένα DBMS. Κάθε DBMS ουσιαστικά δέχεται και εκτελεί ένα σετ εντολών SQL για να διαχειριστεί τα δεδομένα του (εικόνα 3.6). Η MySQL είναι μία διανομή (έκδοσή) της γλώσσας SQL.

```
hg$
hg$ mysql
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 8
Server version: 8.0.17 MySQL Community Server - GPL

Copyright (c) 2000, 2018, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> USE test;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> SHOW tables;
+----------------------------------------------+
| Tables_in_test                               |
+----------------------------------------------+
```
Εικόνα 3.6 MySQL γραμμή εντολών.

Ο MySQL Server λειτουργεί σε μοντέλο πελάτη - εξυπηρετητή ή ενσωματωμένα συστήματα. Το λογισμικό βάσης δεδομένων MySQL είναι ένα σύστημα πελάτη - εξυπηρετητή που αποτελείται από ένα διακομιστή SQL πολλαπλών νημάτων που υποστηρίζει πολλά διαφορετικά προγράμματα και βιβλιοθήκες πελάτη, εργαλεία διαχείρισης και ένα ευρύ φάσμα διασυνδέσεων προγραμματισμού εφαρμογών (APIs). [22]

3.3.3 SQLite

Το SQLite είναι ένα σύστημα διαχείρισης σχεσιακών βάσεων δεδομένων που περιέχεται σε

μια C προγραμματιστική βιβλιοθήκη. Είναι η πιο δημοφιλής επιλογή ως ενσωματωμένη βάση δεδομένων για τοπική αποθήκευση ή αποθήκευση πελάτη σε λογισμικό εφαρμογής όπως ο φυλλομετρητής ή άλλου είδους εφαρμογών. Είναι η πιο ευρέως αναπτυσσόμενη βάση δεδομένων που χρησιμοποιείται, συνδέεται με πολλές γλώσσες και η πηγαία γλώσσα της είναι το public domain. [23]

Το πρόγραμμα της εφαρμογής καλεί συναρτήσεις μέσα από απλές διεργασίες για να χρησιμοποιήσει το SQLite, μειώνοντας την καθυστέρηση ενώ η λειτουργικότητα είναι πιο αποτελεσματική σε σχέση με την διαδιεργασική επικοινωνία. Το SQLite υλοποιεί το μεγαλύτερο μέρος του προτύπου SQL-92 για το SQL, αλλά του λείπουν κάποια χαρακτηριστικά. Υποστηρίζει όμως την σύνταξη SQL, τις συναλλαγές και τα έτοιμα ερωτήματα. Έχει συνδέσεις για έναν μεγάλο αριθμό γλωσσών προγραμματισμού, που συμπεριλαμβάνουν: Basic, C/C++, C#, Java, JavaScript, Python, Perl, PHP, Ruby κ.ά. Επιπλέον, η βάση δεδομένων απαιτεί περιορισμένη μνήμη κατά το χρόνο εκτέλεσης (περίπου. 250 KByte) που αποδεικνύει ότι το SQLite είναι καλά προσαρμοσμένο σε ενσωματωμένα συστήματα και περιλαμβάνεται επίσης στα: BlackBerry 10, Windows Phone 8, iOS, Android, Maemo, MeeGo, webOS, NetBSD, OpenBSD, FreeBSD, illumos. [23]

Το Android είναι ένα από τα λειτουργικά συστήματα που υποστηρίζουν το SQLite και για το λόγο αυτό, είναι ενσωματωμένο σε κάθε συσκευή Android. Όταν χρησιμοποιείται μια βάση δεδομένων δεν απαιτεί μια διαδικασία εγκατάστασης ή διαχείριση της βάσης δεδομένων. Το μόνο που έχει να κάνει ο διαχειριστής είναι να καθορίσει τις δηλώσεις SQL για την δημιουργία και την ενημέρωση της βάσης δεδομένων. Έπειτα, η βάση δεδομένων διαχειρίζεται αυτόματα τα δεδομένα από την πλατφόρμα του Android. Η πρόσβαση σε μια βάση δεδομένων SQLite περιλαμβάνει την πρόσβαση στο σύστημα αρχείων το οποίο μπορεί να έχει καθυστέρηση. Ως εκ τούτου, συνιστάται να εκτελεστούν οι λειτουργίες βάσης δεδομένων ασύγχρονα. [23]

Η βάση του κώδικα SQLite υποστηρίζεται από μια διεθνή ομάδα προγραμματιστών πλήρους απασχόλησης. Οι προγραμματιστές συνεχίζουν να επεκτείνουν τις δυνατότητες του SQLite και να ενισχύουν την αξιοπιστία και την απόδοσή της, διατηρώντας παράλληλα

Εικόνα 3.7 SQLite λογότυπο.

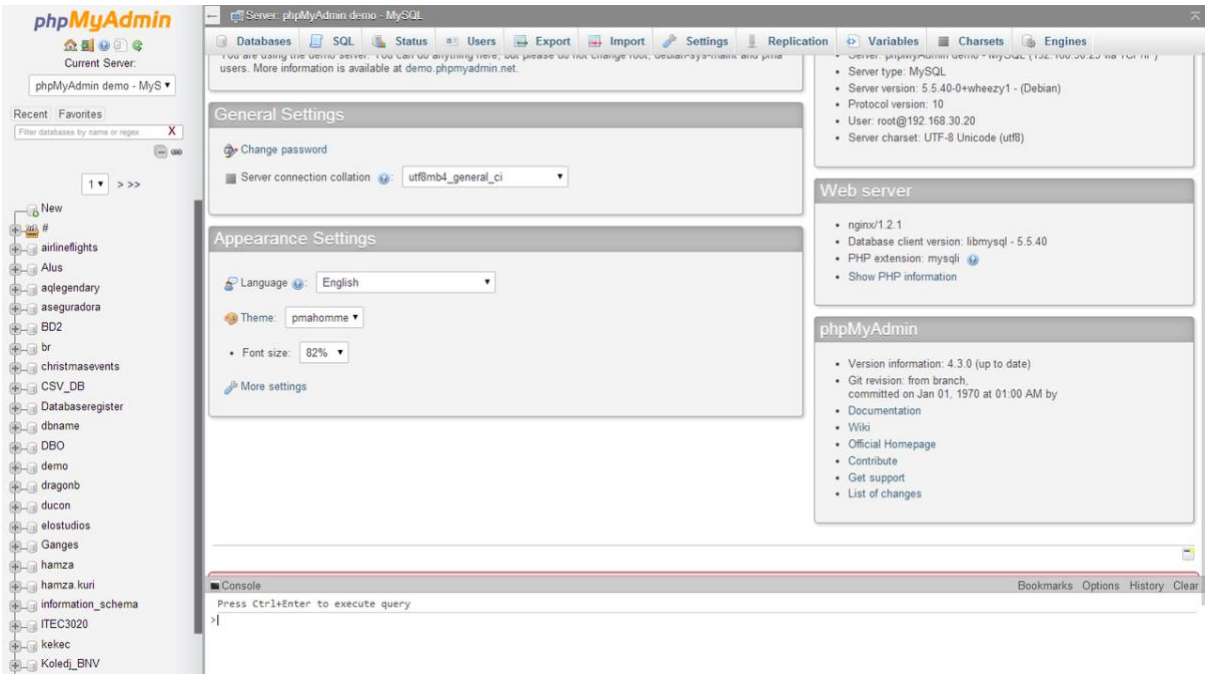
συμβατότητα με το πρότυπο SQL, και τη μορφή αρχείου βάσης δεδομένων.

3.3.4 Εργαλείο phpMyAdmin

Το λογισμικό phpMyAdmin (εικόνα 3.8) όπως προαναφέρθηκε, υποστηρίζει ένα μεγάλο εύρος από τις λειτουργίες της MySQL και είναι ένα ώριμο έργο με σταθερή και ευέλικτη βάση κώδικα. Το phpMyAdmin είναι ουσιαστικά μια διαδικτυακή εφαρμογή που επιτρέπει την διαχείριση (με οπτική διεπαφή) βάσεων δεδομένων MySQL βασισμένες σε έναν εξυπηρετητή (αγγλικά: server). Υπάρχει η δυνατότητα να δημιουργηθούν, να ενημερωθούν, να διαγραφθούν, να αλλάξουν, να εισαχθούν και να εξαχθούν πίνακες βάσεων δεδομένων

χρησιμοποιώντας αυτό το λογισμικό. Το phpMyAdmin υποστηρίζει επίσης ένα ευρύ φάσμα λειτουργιών όπως διαχείριση σχέσεων, πινάκων, στηλών, ευρετηρίων, αδειών, χρηστών κ.λ.π στην MySQL και την MariaDB. Αυτές οι λειτουργίες μπορούν να εκτελεστούν μέσω διεπαφής χρήστη, ενώ υπάρχει η δυνατότητα εκτέλεσης οποιασδήποτε δήλωσης SQL. Κάποια επιπλέον χαρακτηριστικά που υποστηρίζει το λογισμικό αυτό είναι τα παρακάτω:

- Εισαγωγή δεδομένων από CSV και SQL.
- Εξαγωγή δεδομένων σε διάφορες μορφές: CSV, SQL, XML, PDF, ISO / IEC 26300 - OpenDocument Text και Spreadsheet, Word και άλλα.
- Δημιουργία γραφικών της διάταξης της βάσης δεδομένων σε διάφορες μορφές.
- Διαχείριση πολλαπλών διακομιστών.
- Δημιουργία σύνθετων ερωτημάτων χρησιμοποιώντας το ερώτημα-με-παράδειγμα (QBE).
- Αναζήτηση παγκοσμίως σε μια βάση δεδομένων ή ένα υποσύνολο αυτής.
- Μετατροπή αποθηκευμένων δεδομένων σε οποιαδήποτε μορφή χρησιμοποιώντας ένα σύνολο προκαθορισμένων λειτουργιών, όπως η εμφάνιση δεδομένων BLOB ως εικόνα ή σύνδεσμος λήψης. [20][21]



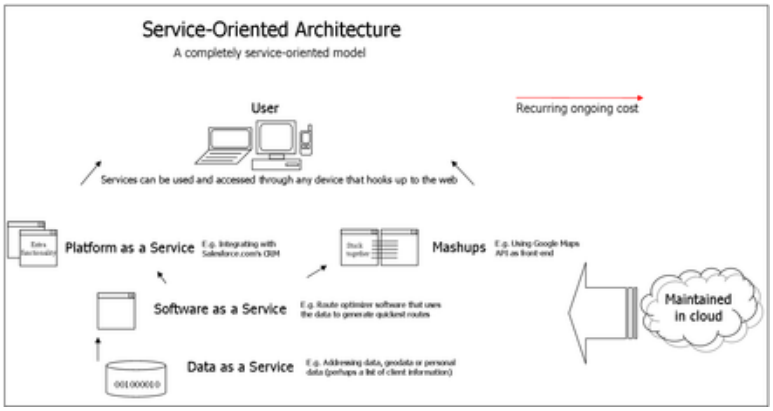
Εικόνα 3.8 Αποτύπωμα οθόνης από την αρχική οθόνη του phpMyAdmin.

3.4 Web Services

Μια διαδικτυακή υπηρεσία (αγγλικά: web service) είναι μια συλλογή ανοιχτών πρωτοκόλλων και προτύπων που χρησιμοποιούνται για την ανταλλαγή δεδομένων μεταξύ εφαρμογών ή συστημάτων. Οι εφαρμογές λογισμικού που είναι γραμμένες σε διάφορες γλώσσες προγραμματισμού και εκτελούνται σε διαφορετικές πλατφόρμες μπορούν να χρησιμοποιούν υπηρεσίες διαδικτύου για την ανταλλαγή δεδομένων μέσω δικτύων υπολογιστών όπως το διαδίκτυο με τρόπο παρόμοιο με την επικοινωνία μεταξύ διεργασιών σε έναν μόνο υπολογιστή. Αυτή η διαλειτουργικότητα (π.χ. μεταξύ Java και Python, ή εφαρμογών Windows και Linux) οφείλεται στη χρήση ανοιχτών προτύπων. Είναι στην πραγματικότητα μια υπηρεσία που προσφέρεται από μια ηλεκτρονική συσκευή σε άλλη, η οποία επικοινωνεί μεταξύ τους μέσω

του διαδικτύου (World Wide Web) ή ένας διακομιστής που εκτελείται σε μια συσκευή υπολογιστή, ακούει αιτήματα σε μια συγκεκριμένη θύρα μέσω δικτύου και εξυπηρετεί έγγραφα Ιστού (HTML, JSON, XML, εικόνες). [24] [25]

Μια υπηρεσία Ιστού είναι μια μονάδα διαχειριζόμενου κώδικα που μπορεί να γίνει επίκληση από απόσταση χρησιμοποιώντας HTTP. Δηλαδή μπορεί να ενεργοποιηθεί χρησιμοποιώντας αιτήματα HTTP. Οι υπηρεσίες Ιστού επιτρέπουν την έκθεση της λειτουργικότητας του υπάρχοντος κώδικα μέσω του δικτύου. Μόλις εκτεθεί στο δίκτυο, άλλες εφαρμογές μπορούν να χρησιμοποιήσουν τη λειτουργικότητα του προγράμματος. Στην πράξη παρέχουν συνήθως μια αντικειμενοστραφή διεπαφή βασισμένη στο διαδίκτυο σε έναν διακομιστή βάσης δεδομένων, που χρησιμοποιείται για παράδειγμα από έναν άλλο διακομιστή ιστού ή από μια εφαρμογή για κινητά, που παρέχει μια διεπαφή στον τελικό χρήστη. Πολλοί οργανισμοί που παρέχουν δεδομένα σε μορφοποιημένες σελίδες HTML θα παρέχουν επίσης αυτά τα δεδομένα στον διακομιστή τους ως XML ή JSON, συχνά μέσω μιας υπηρεσίας Web για να επιτρέπουν τη διανομή, για παράδειγμα το Wikipedia's Export. Μια άλλη εφαρμογή που προσφέρεται στον τελικό χρήστη μπορεί να είναι ένα mashup, όπου ένας διακομιστής Web καταναλώνει πολλές υπηρεσίες ιστού σε διαφορετικά μηχανήματα και συγκεντρώνει το περιεχόμενο σε μία διεπαφή χρήστη (εικόνα 3.9). [24] [25]



Εικόνα 3.9 Σχήμα της λειτουργίας των υπηρεσιών ιστού.

3.4.1 SOAP vs REST Web Services

Το SOAP σημαίνει Simple Object Access Protocol ενώ το REST σημαίνει Representational State Transfer. Το SOAP είναι ένα πρωτόκολλο που σχεδιάστηκε πριν από το REST. Η κύρια ιδέα πίσω από το σχεδιασμό του SOAP ήταν να διασφαλιστεί ότι προγράμματα που βασίζονται σε διαφορετικές πλατφόρμες και γλώσσες προγραμματισμού θα μπορούσαν να ανταλλάσσουν δεδομένα με εύκολο τρόπο. Το REST σχεδιάστηκε ειδικά για εργασία με στοιχεία μέσων, αρχεία ή ακόμη και αντικείμενα σε μια συγκεκριμένη συσκευή υλικού. Οποιαδήποτε υπηρεσία ιστού που ορίζεται στις αρχές του REST μπορεί να ονομάζεται υπηρεσία ιστού RestFul. Μια τέτοια υπηρεσία θα χρησιμοποιούσε τα κανονικά ρήματα HTTP των GET, POST, PUT και DELETE για εργασία με τα απαιτούμενα στοιχεία.

Κάθε τεχνική έχει τα δικά της πλεονεκτήματα και μειονεκτήματα. Ως εκ τούτου, είναι πάντα καλό να γίνεται κατανοητό σε ποιες περιπτώσεις πρέπει να χρησιμοποιείται ο κάθε σχεδιασμός. Παρόλα αυτά ως καλύτερη επιλογή θεωρείται το REST διότι εκτός από τη χρήση HTTP για απλότητα, προσφέρει μια σειρά από άλλα πλεονεκτήματα σε σχέση με το SOAP. Επιτρέπει μια μεγαλύτερη ποικιλία μορφών δεδομένων, ενώ το SOAP επιτρέπει μόνο XML. Σε συνδυασμό με το JSON (το οποίο συνήθως λειτουργεί καλύτερα με δεδομένα και

προσφέρει ταχύτερη ανάλυση), το REST θεωρείται γενικά ευκολότερο στην εργασία. Είναι το πρότυπο, το οποίο χρησιμοποιήθηκε στην ανάπτυξη της εφαρμογής CarBook στα πλαίσια της παρούσας διπλωματικής εργασίας, λόγω της ταχύτερης ανάλυσης που προσφέρει.
Οι υπηρεσίες REST είναι αποδοτικότερο να χρησιμοποιούνται στις ακόλουθες περιπτώσεις:

- Περιορισμένοι πόροι και εύρος ζώνης: Δεδομένου ότι τα μηνύματα SOAP έχουν βαρύτερο περιεχόμενο και καταναλώνουν πολύ μεγαλύτερο εύρος ζώνης, το REST θα πρέπει να χρησιμοποιείται σε περιπτώσεις όπου το εύρος ζώνης δικτύου είναι περιορισμένο.
- Ανιθαγένεια: Εάν δεν υπάρχει ανάγκη διατήρησης της κατάστασης πληροφοριών από το ένα αίτημα στο άλλο, τότε το REST πρέπει να χρησιμοποιείται. Εάν απαιτείται μια σωστή ροή πληροφοριών όπου ορισμένες πληροφορίες από ένα αίτημα πρέπει να ρέουν σε άλλη, τότε το SOAP είναι πιο κατάλληλο για το σκοπό αυτό.
- Caching: Όταν υπάρχει ανάγκη για προσωρινή αποθήκευση πολλών αιτημάτων, τότε το REST είναι η τέλεια λύση. Κατά καιρούς, οι πελάτες θα μπορούσαν να ζητήσουν τον ίδιο πόρο πολλές φορές. Αυτό μπορεί να αυξήσει τον αριθμό των αιτημάτων που αποστέλλονται στον διακομιστή. Με την εφαρμογή μιας προσωρινής μνήμης, τα πιο συχνά αποτελέσματα των ερωτημάτων μπορούν να αποθηκευτούν σε μια ενδιάμεση τοποθεσία.
- Ευκολία κωδικοποίησης: Η κωδικοποίηση REST Services και η επακόλουθη εφαρμογή είναι πολύ ευκολότερη από το SOAP. Επομένως, εάν απαιτείται μια λύση γρήγορης λύσης για υπηρεσίες ιστού, τότε το REST είναι ο τρόπος που αξίζει να ακολουθηθεί.

Οι υπηρεσίες SOAP είναι αποδοτικότερο να χρησιμοποιούνται στις ακόλουθες περιπτώσεις:

- Ασύγχρονη επεξεργασία και επακόλουθη επίκληση: Εάν υπάρχει η απαίτηση ότι ο πελάτης χρειάζεται ένα εγγυημένο επίπεδο αξιοπιστίας και ασφάλειας, τότε το νέο πρότυπο SOAP του SOAP 1.2 παρέχει πολλές πρόσθετες δυνατότητες, ειδικά όταν πρόκειται για ασφάλεια.
- Ένα επίσημο μέσο επικοινωνίας: Εάν τόσο ο πελάτης όσο και ο διακομιστής έχουν συμφωνία σχετικά με τη μορφή ανταλλαγής, τότε το SOAP 1.2 παρέχει τις άκαμπτες προδιαγραφές για αυτόν τον τύπο αλληλεπίδρασης.
- Stateful λειτουργίες: Εάν η εφαρμογή έχει την απαίτηση η κατάσταση να πρέπει να διατηρηθεί από το ένα αίτημα στο άλλο, τότε το πρότυπο SOAP 1.2 παρέχει τη δομή WS * για την υποστήριξη τέτοιων απαιτήσεων.

Κεφάλαιο 4: Υλοποίηση εφαρμογής Carbook για Android

Μετά την ανάλυση κάποιων βασικών εννοιών που αναφέρονται στην παρούσα διπλωματική εργασία, σειρά έχει η παρουσίαση της διαδικασίας που ακολουθήθηκε για την ανάπτυξη της εφαρμογής Carbook από τα πρώτα στάδια της έρευνας μέχρι την λειτουργική εφαρμογή. Για τον σκοπό αυτό, στα επόμενα υποκεφάλαια αρχικά γίνεται περιγραφή της έρευνας και της μελέτης περιπτώσεων της πραγματικής ζωής σχετικά μετά αυτοκίνητα που επισκευάζονται από τα αντίστοιχα συνεργεία. Παράλληλα γίνεται μια σύντομη παρουσίαση των δεδομένων που συλλέχθηκαν από την έρευνα στο διαδίκτυο και τις σύντομες συνεντεύξεις από καταρτισμένο προσωπικό συνεργείων και αντιπροσωπειών και από μηχανικούς αυτοκινήτων. Στην συνέχεια γίνεται ανάλυση της αρχιτεκτονικής που έχει η εφαρμογή καθώς και της βάσης δεδομένων που προέκυψε από τα δεδομένα της έρευνας και της διαδικασίας για την δημιουργία της. Τέλος ακολουθεί η διαδικασία προγραμματισμού του συστήματος διαχείρισης επισκευών αυτοκινήτων Carbook στο IDE περιβάλλον Android Studio και η σύνδεση της βάσης με την εφαρμογή μέσω web services.

Εικόνα 4.1 Γλώσσες προγραμματισμού

4.1 Έρευνα και μελέτη πραγματικής ζωής

Αυτό είναι το πρώτο στάδιο που απαιτείται για να γίνει πραγματικότητα το σύστημα Carbook. Ο λόγος που υφίσταται αυτή η απαίτηση είναι ότι η εφαρμογή που παρουσιάζεται εδώ απαιτεί ένα μεγάλο όγκο δεδομένων για τα οχήματα που διαχειρίζεται τα οποία χρειάζεται να αποθηκεύονται σε μια βάση δεδομένων. Η κατασκευή της εφαρμογής βασίζεται στην δημιουργία ο ενός ορθού συστήματος σχεσιακής βάσης δεδομένων και κατ' επέκταση στην λεπτομερή μελέτη και έρευνα περιπτώσεων που συμβαίνουν στην καθημερινότητα.

4.1.1 Μελέτη πραγματικής ζωής

Το πρώτο βήμα που έγινε για να είναι δυνατή η συλλογή δεδομένων είναι κάποιες σύντομες συνεντεύξεις σε δύο μηχανικούς που δουλεύουν σε αντιπροσωπείες αυτοκινήτων και σε μια υπάλληλο σε αντιπροσωπεία που παρέχει γραμματειακή υποστήριξη. Πιο συγκεκριμένα

υπήρξε επικοινωνία με τον κύριο Γιάννη που είναι κορυφαίος μηχανικός σε αντιπροσωπεία της αυτοκινητοβιομηχανίας Fiat, την κυρία Ελισάβετ η οποία έχει τα καθήκοντα της στην γραμματεία της ίδιας αντιπροσωπείας πολλά χρόνια και τέλος με τον Γιάννη ο οποίος είναι (νέος σχετικά στον χώρο) μηχανικός σε αντιπροσωπεία της αυτοκινητοβιομηχανίας Volkswagen.

Σε αυτές τις συνεντεύξεις δόθηκε η ευκαιρία να γίνουν ερωτήσεις σχετικά με την αυτοκίνηση σε πολλά επίπεδα, ώστε οι έμπειροι επαγγελματίες στον χώρο να έχουν και κατατοπιστικό ρόλο στην έρευνα και την μελέτη που είναι αναγκαία. Υπήρξαν γενικές ερωτήσεις για τον τρόπο λειτουργίας των συνεργείων και κάποιες ειδικού τύπου πάνω στα διάφορα μέρη των αυτοκινήτων, στις εργασίες που γίνονται σε αυτά και την οργάνωση των πληροφορίων που προκύπτουν είτε από τους πελάτες είτε από τα οχήματα. Κατά την διάρκεια της μελέτης που έγινε προέκυψαν επιπλέον απορίες και πραγματοποιήθηκαν συμπληρωματικά διευκρινήσεις. Παρακάτω ακολουθούν τα ερωτήματα που τέθηκαν καθώς και οι απαντήσεις που δόθηκαν από τους δύο μηχανικούς και την γραμματέα.

1. **Ποια είναι η τυπική διαδικασία που ακολουθεί ένας μηχανικός από την έναρξη μέχρι την ολοκλήρωση της επισκευής ενός αυτοκινήτου;**

«Αρχικά ελέγχουμε το βιβλίο επισκευών για να δούμε τι εργασίες έχουν γίνει ή υπολείπονται, στην συνέχεια γίνεται φυσικός έλεγχος στο όχημα (ανάλογα φυσικά τι εργασία θέλει ο πελάτης να γίνει). Σε κάποιες περιπτώσεις γίνεται προληπτικός έλεγχος σε μέρη του αυτοκινήτου που λόγω της εργασίας που είναι να γίνει αφαιρείται κάποιο εξάρτημα που δίνει πρόσβαση σε άλλα που είναι πιο δύσκολο να τα κοιτάξουμε υπό άλλες συνθήκες. Πάντα οι επισκευές γίνονται σύμφωνα με τις οδηγίες της αυτοκινητοβιομηχανίας. Έπειτα αφού προμηθευτούμε τα απαραίτητα ανταλλακτικά πραγματοποιούμε την επισκευή και σημειώνουμε στο βιβλίο επισκευών ότι εργασία και έλεγχος έγινε. Σε περίπτωση που ο πελάτης δεν κρατάει τέτοιο βιβλίο τον συμβουλεύουμε να το κάνει. Σε γενικές γραμμές αυτή είναι η γενική διαδικασία.» Απάντησε ο μηχανικός της Fiat κύριος Γιάννης.

«Στην αντιπροσωπία που δουλεύω (της Volkswagen) κύρια αρμοδιότητα μου είναι να κάνω το service ή τις επισκευές στα οχήματα που μου αναθέτουν. Οπότε συνήθως μου δίνεται μια λίστα με τις ζητούμενες εργασίες και έπειτα από έλεγχο που πραγματοποιώ ο ίδιος ή οι συνάδελφοί μου κάνω την επισκευή ή την συντήρηση. Υπάρχουν δύο κύριες κατηγορίες service, είναι το μικρό και το μεγάλο service. Σε αυτές τις κατηγορίες γίνεται η συντήρηση και η αντικατάσταση εξαρτημάτων που χρειάζονται αλλαγή ή επισκευή ταχτικά. Το μικρό service είναι στην ουσία η προγραμματισμένη ετήσια συντήρηση η οποία σύμφωνα με τους περισσότερους κατασκευαστές πρέπει να γίνεται με τη συμπλήρωση ενός έτος ή των 10.000 χιλιομέτρων. Πέρα από τον γενικό έλεγχο της κατάστασης του οχήματος, γίνεται αντικατάσταση λιπαντικού κινητήρα και φίλτρου λαδιού. Στα μεγάλα service ή ανά δύο μεγάλα συνηθίζεται να γίνεται και αντικατάσταση των μπουζί. Επίσης, υπάρχει κι άλλη μία προγραμματισμένη αντικατάσταση, η οποία βέβαια δεν συμπεριλαμβάνεται σε κάθε μεγάλο service και είναι η αντικατάσταση του ιμάντα χρονισμού αν δεν χρησιμοποιείται καδένα. Ασφαλώς αυτά τα διαστήματα μεταβάλλονται ανάλογα με το κάθε μοντέλο, για αυτό είναι καλύτερο να συμβουλευόμαστε το εγχειρίδιο χρήσης.» Απάντησε ο μηχανικός της Volkswagen Γιάννης.

2. **Υπάρχουν συγκεκριμένα διαστήματα που καθορίζουν κάθε πότε και ποιες εργασίες γίνονται;**

«Συνήθως αν πρόκειται για αμάξι πιο καινούριο ή για αμάξι που τηρούνται πιστά τα διαστήματα που απαιτούνται να γίνονται οι έλεγχοι, η συντήρηση και οι επισκευές πραγματοποιούνται τότε με βάση τα επίσημα διαστήματα που έχει ορίσει ο κάθε κατασκευαστής (που συνήθως υπάρχουν στο εγχειρίδιο χρήσης του οχήματος). Στα πιο καινούρια αμάξια είναι σημαντικό να τηρούνται τα διαστήματα για να μην χαθεί η εγγύηση του κατασκευαστή. Στα πολύ παλιά μοντέλα πολλές φορές αμελούνται κάποιες εργασίες ή δεν γίνονται στην ώρα τους οπότε πράττουμε ανάλογα την περίπτωση και φυσικά ανάλογα την διαθεσιμότητα των ανταλλακτικών.» Απάντησε η γραμματέας της Fiat κυρία Ελισσάβετ.

«Κάποιες κατηγορίες εξαρτημάτων απαιτούν συχνότερη αντικατάσταση ή επισκευή. Κάθε κατηγορία μερών αυτοκινήτων έχουν συγκεκριμένο χρόνο που απαιτείται συντήρηση, για παράδειγμα ο ιμάντας χρονισμού συνήθως θέλει αντικατάσταση κάθε 90.000 χιλιόμετρα πάνω κάτω ανάλογα το αμάξι. Όμως όσο περνάει ο καιρός, οι εταιρίες που παράγουν τα εξαρτήματα των αυτοκινήτων (δεν παράγουν οι αυτοκινητοβιομηχανίες ένα μεγάλο ποσοστό των εξαρτημάτων) αλλάζουν τον τρόπο κατασκευής και επειδή η τεχνολογία και τα υλικά αναπτύσσονται, αλλάζει ο χρόνος ζωής τους οπότε σε πολλές περιπτώσεις αλλάζουν τα διαστήματα επισκευής και αντικατάστασης. Αυτό συμβαίνει συνήθως όταν ένα μοντέλο αυτοκινήτου κυκλοφορεί στην αγορά αρκετά χρόνια.» Απάντησε ο μηχανικός της Fiat κύριος Γιάννης.

«Η αυτοκινητοβιομηχανία έχει καθορίσει συγκεκριμένα χρονικά πλαίσια στα οποία γίνονται έλεγχοι και αντικαταστάσεις των εξαρτημάτων, των υγρών. Με βάση αυτά γίνονται οι εργασίες, ειδικά στα πιο καινούρια οχήματα.» Απάντησε ο μηχανικός της Volkswagen Γιάννης.

3. Τι πληροφορίες κρατούνται στο βιβλίο επισκευών των αυτοκινήτων που αναλαμβάνετε;

«Οι πληροφορίες που καταγράφονται είναι οι εξής: χιλιόμετρα στα οποία γίνεται η εργασία, ημερομηνία επίσκεψης στο συνεργείο, αριθμός χιλιομέτρων στα οποία θα γίνει η επόμενη επίσκεψη και τέλος οι εργασίες που γίνονται στην τρέχουσα επίσκεψη.» Η απάντηση αντλήθηκε από όλους τους ερωτηθέντες οι οποίοι σαν απάντηση έδειξαν τα προσωπικά τους βιβλία επισκευών των αυτοκινήτων τους με τις πληροφορίες που σημειώνουν.

4. Από που μπορώ να μάθω τα διαστήματα όπου γίνονται οι προληπτικοί έλεγχοι, η συντήρηση και οι επισκευές;

«Αυτή η πληροφορία μπορεί να γίνει γνωστή από την ίδια την αυτοκινητοβιομηχανία που παράγει τα οχήματα είτε απευθείας από το εγχειρίδιο χρήσης του οχήματος, όπως για παράδειγμα στα αυτοκίνητα της Fiat, είτε από την αντιπροσωπεία που πουλάει τα οχήματα και πρέπει ο πελάτης να γνωρίζει ποιες είναι οι υποχρεωτικές εργασίες που πρέπει να γίνονται τα πρώτα χρόνια για να διατηρήσει την εγγύηση που προσφέρει η εταιρία.» Απάντησε η γραμματέας της Fiat κυρία Ελισσάβετ. Οι δύο μηχανικοί συμφώνησαν με αυτή την απάντηση.

5. Κατά την άποψη σας, θα διευκόλυνε την όλη διαδικασία εάν εκτός από το φυσικό βιβλίο επισκευών, οι πελάτες έχουν στην διάθεσή τους μια εφαρμογή στο κινητό για αυτό το σκοπό;

«Εμένα προσωπικά, επειδή δεν τα πάω καλά με τα κινητά και την νέα τεχνολογία θα με δυσκόλευε και σίγουρα θα προτιμούσα τον κλασσικό τρόπο. Σίγουρα όμως θα ήταν χρήσιμη

επιλογή για τους ιδιοκτήτες των οχημάτων που θέλουν να έχουν ένα αντίγραφο των δεδομένων αυτών και να μπορούν να ρίξουν μια γρήγορη ματιά χωρίς να ψάχνουν το βιβλίο.» Απάντησε ο μηχανικός της Fiat κύριος Γιάννης.

«Πιστεύω ότι είναι ένα βήμα στην σωστή κατεύθυνση γιατί στις μέρες όλα τείνουν να γίνονται ηλεκτρονικά και για πολλούς θα είναι πολύ χρήσιμη επιλογή.» Απάντησε η γραμματέας της Fiat κυρία Ελισσάβετ.

«Είναι πολύ ενδιαφέρον ιδέα και χρήσιμη. Κάτι τέτοιο θα μας έλυνε τα χέρια σαν κάτοχους των αυτοκινήτων.» Απάντησε ο μηχανικός της Volkswagen Γιάννης.

6. Ποια ήδη συνεργείων υπάρχουν; Υπάρχουν εξιδεικευμένα σε κάποια συγκεκριμένα μέρη του αυτοκινήτου;

«Τα κύρια ήδη συνεργείων είναι τρία, αυτά που απασχολούνται με τα μηχανικά θέματα, με τα ηλεκτρολογικά και είναι επίσης τα φανοποιεία. Βεβαία μια αντιπροσωπεία αναλαμβάνει να τα καλύψει όλα αυτά. Τα πιο μικρά συνεργεία συνήθως είναι αυτά που κάνουν κάποια από αυτά ή ακόμα εξειδικεύονται σε συγκεκριμένα θέματα. Λόγω του γεγονότος ότι ένα όχημα έχει μεγάλο αριθμό συστημάτων και εξαρτημάτων υπάρχουν αρκετές κατηγορίες συνεργείων. Οι πιο γνωστές είναι: συνεργεία που επισκευάζουν φρένα, εκείνα που αναλαμβάνουν τα συστήματα εξατμίσεων, τα συστήματα αναρτήσεων πολλές φορές μαζί με την αλλαγή ελαστικών, ευθυγράμμισης και ζυγοστάθμισής των τροχών (το τελευταίο μπορεί να αποτελέσει και ξεχωριστή κατηγορία), συνεργεία που αντικαθιστούν τα τζάμια και τους καθρέπτες του αυτοκίνητου και επίσης εκείνα που ειδικεύονται στην επισκευή και το χτίσιμο των μηχανών των αυτοκινήτων.» Απάντησε η γραμματέας της Fiat κυρία Ελισσάβετ, σε συμφωνία όμως με τις απαντήσεις των δύο μηχανικών.

7. Ποιες πληροφορίες πρέπει να γνωρίζει οπωσδήποτε ο κάθε κάτοχος για το αυτοκίνητο του ώστε να είναι αναγνωρίσιμο σαν μοντέλο και σαν όχημα γενικότερα;

Ένας ιδιοκτήτης, για να μπορεί να αναγνωρίζει το όχημα του πρέπει οπωσδήποτε να θυμάται τον αριθμό πινακίδας και φυσικά το χρώμα του σασί του αυτοκινήτου. Ένας χρήσιμος κωδικός για τα service είναι ο αριθμός πλαισίου που φαίνεται από το ίδιο το αυτοκίνητο διότι με αυτό τον αριθμό μπορούν να αναζητηθούν και να βρεθούν πιο εύκολα και αποδοτικά τα πιθανά ανταλλακτικά, τα οποία ακόμα και σε ίδια μοντέλα μπορεί να διαφέρουν ανάλογα τις εκδόσεις και τα χαρακτηριστικά των κινητήρων. Για να γίνει η αναγνώριση του μοντέλου και όχι του συγκεκριμένου οχήματος του κατόχου είναι απαραίτητο να γνωρίζουμε τα χαρακτηριστικά του, δηλαδή τα κυβικά του κινητήρα, οι ίπποι που αποδίδει, ο τύπος καυσίμου και κάποια πιο γενικά πράγματα όπως τι χρονιάς είναι, ποιο μοντέλο και ποιας μάρκας. Αυτή είναι η ίδια απάντηση που λήφθηκε από όλα τα πρόσωπα τα οποία συμμετείχαν στην συνέντευξη.

8. Ποιες είναι σε γενικές γραμμές οι εργασίες που μπορούν να πραγματοποιηθούν στα αυτοκίνητα (δηλαδή για κάθε ένα από τα συστήματα-μέρη του);

«Τα είδη που μπορούμε να κατατάξουμε τις εργασίες είναι τα εξής: έχουμε έλεγχο συστημάτων ή εξαρτημάτων, αντικατάσταση εξαρτημάτων ή επισκευή (σε περιπτώσεις που με μικρές πατέντες μπορεί να λειτουργήσει για λίγο καιρό παραπάνω ένα σύστημα ή εξάρτημα), σε κάποιες περιπτώσεις μπορεί να γίνει καθαρισμός κάποιου εξαρτήματος όπως για παράδειγμα σε κάποια φίλτρα του αυτοκινήτου ώστε να κερδίσει ο κάτοχος λίγα χιλιόμετρα λειτουργίας πριν από κάποιο κοντινό μεγάλο service και τέλος μπορεί να γίνει

προσθήκη υγρών (εκτός από αλλαγή) σε περιπτώσεις που υπάρχει μικρή διαρροή.» Απάντησε η γραμματέας της Fiat κυρία Ελισσάβετ.

Εικόνα 4.2 Οι πιο δημοφιλείς αυτοκινητοβιομηχανίες ταξινομημένες με βάση τις θυγατρικές εταιρίες στις οποίες ανήκουν.

Εκτός από τις παραπάνω ερωτήσεις υπήρξαν διάφορες διευκρινήσεις που έγιναν κατά την διάρκεια της έρευνας και μελέτης των αυτοκινήτων μαζί με συζήτηση πάνω σε στοιχεία που συλλέχτηκαν από το διαδίκτυο ώστε να είναι επιβεβαιωμένη η ορθότητα τους. Αυτά τα στοιχεία αφορούν κυρίως την αναζήτηση εγχειρίδιων χρήσης για τα μοντέλα που θα είναι καταχωρημένα στο σύστημα της βάσης δεδομένων που αναπτύσσεται, καθώς και κάποιες διευκρινήσεις πάνω σε εργασίες που αναφέρονται στο «πρόγραμμα συντήρησης» των κατασκευαστών. Έγινε επιπλέον συμπληρωματική διερεύνηση πάνω στα ερωτήματα που τέθηκαν ώστε να γίνουν πλήρως κατανοητές οι απαντήσεις και για τις πιθανών πληροφορίες που μπορούν να βρεθούν. Διαδικτυακή έρευνα έγινε επίσης ώστε να βρεθούν όλες οι εκδόσεις των μοντέλων αυτοκινήτων που θα εισαχθούν στην βάση δεδομένων ώστε να είναι σωστά τα στοιχεία που θα αποθηκευτούν.

Στο επόμενο υποκεφάλαιο θα παρουσιαστούν τα δεδομένα και τα συμπεράσματα σχετικά με τις επισκευές των αυτοκινήτων και τα συνεργεία-αντιπροσωπείες.

4.1.2 Παρουσίαση των δεδομένων για τα διαστήματα επισκευών

Σε αυτό το μέρος της εργασίας μετά από εκτεταμένη διαδικτυακή έρευνα πάντα παράλληλα με τις συμβουλές από τους μηχανικούς έχουν συλλεχθεί τα απαραίτητα δεδομένα για τα διαστήματα της συντήρησης με βάση τους κατασκευαστές, δηλαδή είναι πλέον γνωστό κάθε πόσα χιλιόμετρα και μήνες πρέπει να γίνεται η κάθε εργασία συντήρησης στα επιλεγμένα αυτοκίνητα. Τα μοντέλα που έχουν επιλεγεί είναι τα Nissan Micra K14 (εικόνα 4.3), Nissan Leaf ZE1 (εικόνα 4.4), Fiat Panda Mk4 (319) (εικόνα 4.5), Fiat Tipo (εικόνα 4.6) και Smart Fortwo (C453) (εικόνα 4.7) (με όλες τις εκδόσεις τους) και σκοπός είναι με την πάροδο του χρόνου να προστεθούν ακόμα περισσότερα μοντέλα ώστε να καλύπτει η εφαρμογή ένα μεγάλο εύρος αυτοκινήτων. Παρόλα αυτά για τους σκοπούς της παρούσας εργασίας το δείγμα θα είναι

σχετικά μικρό ώστε να είναι πιο διαχειρίσιμο ως προς τον χρόνο υλοποίησης. Εξάλλου ο κύριος σκοπός του συστήματος που αναπτύσσεται είναι η διαχείριση των πληροφορίων των βιβλίων επισκευών και παράλληλα να είναι γνωστό από την εφαρμογή κάθε πότε πρέπει ο χρήστης να επισκεφτεί το συνεργείο. Παρακάτω ακολουθούν τα επίσημα διαστήματα και οι

Εικόνα 4.3 Nissan Micra τελευταίας γενιάς.

σαφείς οδηγίες από τις αυτοκινητοβιομηχανίες για τις εργασίες που πρέπει να γίνονται για κάθε μοντέλο σε μορφή πινάκων.

Ο παρακάτω πίνακας αφορά το Nissan Micra τελευταίας γενιάς με κινητήρες πετρελαίου (του 2017 μέχρι σήμερα):

Εργασίες συντήρησης		**Διαστήματα συντήρησης**									
Εκτέλεση σε διαστήματα χιλιομέτρων (μίλια) ή μήνα, όποιο από τα δύο συμβεί πρώτο.	km x 1 000 (μίλια x1000) Μήνες	15 (9) 12	30 (18) 24	45 (27) 36	60 (36) 48	75 (45) 60	90 (54) 72	105 (63) 84	120 (72) 96	135 (81) 108	150 (90) 120
Διάκενο βαλβίδας εισαγωγής και εξαγωγής	Σημείωση (1)										
Ζώνη κίνησης	Σημείωση (2)	E	E		E		E		E		E
Λάδι κινητήρα (Χρησιμοποιήστε συνιστώμενο λάδι) ★		A	A	A	A	A	A	A	A	A	A
Φίλτρο λαδιού κινητήρα (Χρησιμοποιήστε γνήσιο φίλτρο λαδιού κινητήρα NISSAN ή ισοδύναμο) ★		A	A	A	A	A	A	A	A	A	A
Ψυκτικό κινητήρα (Χρησιμοποιήστε γνήσιο ψυκτικό κινητήρα NISSAN ή ισοδύναμο στην ποιότητά του) ★	Σημείωση (3)										
Σύστημα ψύξης		E	E	E	E	E	E	E	E	E	E
Γραμμές καυσίμου			E		E		E		E		E
Γραμμές ατμών EVAP (Με μεταλλικό δοχείο)			E		E		E		E		E

Φίλτρο αέρα (τύπου ξηρού χαρτιού) ★			A		A		A		A		A
Φίλτρο αέρα (τύπου παχύρρευστου χαρτιού) ★			A		A		A		A		A
Φίλτρο καυσίμου (τύπου δεξαμενής)	Σημείωση (4)										
Μπουζί (τύπου Iridium-tip) (κινητήρας HR15)							[A]				
Μπουζί (τύπος με νικέλιο) (κινητήρας HR12)			[A]		[A]		[A]		[A]		[A]
Υγρό φρένων και συμπλέκτη (Για επίπεδο & διαρροές)		E	E	E	E	E	E	E	E	E	E
Φρένο & συμπλέκτης, συστήματα εξάτμισης		E	E	E	E	E	E	E	E	E	E
Υγρά φρένων ★			A		A		A		A		A
Ελαστικοί σωλήνες υποπίεσης φρένων, συνδέσεις & βαλβίδα ελέγχου			E		E		E		E		E
Χειροκίνητο κιβώτιο ταχυτήτων (Για διαρροές)		E	E	E	E	E	E	E	E	E	E
Τιμόνι και σύνδεσμος, εξαρτήματα άξονα & ανάρτησης, άξονας έλικα & μπροστινοί άξονες κίνησης ★			E		E		E		E		E
Ευθυγράμμιση τροχού (Εάν είναι απαραίτητο, περιστροφή & ζυγοστάθμιση τροχών)		E	E	E	E	E	E	E	E	E	E
Τακάκια, ρότορες & άλλα εξαρτήματα φρένων ★		E	E	E	E	E	E	E	E	E	E
Επένδυση φρένων, ρότορες & άλλα εξαρτήματα φρένων ★		E	E	E	E	E	E	E	E	E	E
Κλείδωμα, μεντεσέδες & μάνδαλο κουκούλας ★		Λ	Λ	Λ	Λ	Λ	Λ	Λ	Λ	Λ	Λ
Ζώνες ασφαλείας, πόρτες, συσπειρωτήρες, αγκύρια & ρυθμιστές		E	E	E	E	E	E	E	E	E	E
Φρένο ποδιών, χειρόφρενο και συμπλέκτης (για ελεύθερη κίνηση, κίνηση και λειτουργία)		E	E	E	E	E	E	E	E	E	E
Φίλτρο κλιματιστικού ★			A		A		A		A		A

Πίνακας 4.1 Διαστήματα συντήρησης του Nissan Micra με 1,2 και 1,5 λίτρων κινητήρες.

Συντομογραφίες: Α = Αντικατάσταση, Ε = Επιθεώρηση και διόρθωση ή αντικατάσταση, όπως απαιτείται, Λ = Λίπανση

★ Σε πιο απαιτητικές συνθήκες χρήσης του οχήματος γίνονται πιο σύντομα αυτές οι εργασίες.

ΣΗΜΕΙΩΣΕΙΣ:

(1) Δεν απαιτείται περιοδική συντήρηση. Ωστόσο, εάν αυξηθεί ο θόρυβος της βαλβίδας, ελέγξτε το διάκενο της βαλβίδας.

(2) Αντικαταστήστε τον ιμάντα κίνησης εάν βρεθεί κατεστραμμένος.

(3) Χρησιμοποιήστε το γνήσιο ψυκτικό κινητήρα NISSAN (μπλε) ή ισοδύναμο στην ποιότητά του, για να αποφύγετε πιθανή διάβρωση αλουμινίου στο σύστημα ψύξης κινητήρα που προκαλείται από τη χρήση μη γνήσιου ψυκτικού κινητήρα. Πρώτα αντικαταστήστε στα 75

000km (45 000 μίλια) / 60 μήνες και μετά κάθε 45 000km (27 000 μίλια) / 36 μήνες. Εκτελέστε το "E" (Έλεγχος της αναλογίας μείγματος και διόρθωση της αναλογίας μείγματος εάν είναι απαραίτητο) στο μέσο του διαστήματος αντικατάστασης.

(4) Το φίλτρο καυσίμου δεν χρειάζεται συντήρηση.

Ο παρακάτω πίνακας αφορά το Nissan Micra τελευταίας γενιάς με κινητήρες βενζίνης (του 2017 μέχρι σήμερα):

Εργασίες συντήρησης		**Διαστήματα συντήρησης**									
Εκτέλεση σε διαστήματα χιλιομέτρων (μίλια) ή μήνα, όποιο από τα δύο συμβεί πρώτο.	km x 1 000 (μίλια x1000) Μήνες	15 (9) 12	30 (18) 24	45 (27) 36	60 (36) 48	75 (45) 60	90 (54) 72	105 (63) 84	120 (72) 96	135 (81) 108	150 (90) 120
Διάκενο βαλβίδας εισαγωγής και εξαγωγής	Σημείωση (1)										
Ζώνη κίνησης	Σημείωση (2)	Αντικαταστήστε κάθε 90.000 χιλιόμετρα (54.000 μίλια) ή 48 μήνες									
Λάδι κινητήρα (Χρησιμοποιήστε συνιστώμενο λάδι) ★		A	A	A	A	A	A	A	A	A	A
Φίλτρο λαδιού κινητήρα (Χρησιμοποιήστε γνήσιο φίλτρο λαδιού κινητήρα NISSAN ή ισοδύναμο) ★		A	A	A	A	A	A	A	A	A	A
Ψυκτικό κινητήρα (Χρησιμοποιήστε γνήσιο ψυκτικό κινητήρα NISSAN ή ισοδύναμο στην ποιότητά του) ★	Σημείωση (3)										
Σύστημα ψύξης		E	E	E	E	E	E	E	E	E	E
Γραμμές καυσίμου			E		E		E		E		E
Γραμμές ατμών EVAP (Με μεταλλικό δοχείο)			E		E		E		E		E
Φίλτρο αέρα ★		Καθαρίστε κάθε 5.000 χλμ. (3.000 μίλια) - (Λειτουργία DIY)									
			A		A		A		A		A
Φίλτρο καυσίμου (τύπου δεξαμενής)	Σημείωση (4)										
Μπουζί (τύπου Iridium-tip)			A		A		A		A		A
Υγρό φρένων και συμπλέκτη (Για επίπεδο & διαρροές)		E	E	E	E	E	E	E	E	E	E
Φρένο & συμπλέκτης		E	E	E	E	E	E	E	E	E	E
Συστήματα εξάτμισης			E		E		E		E		E
Υγρά φρένων ★			A		A		A		A		A
Ελαστικοί σωλήνες υποπίεσης φρένων, συνδέσεις & βαλβίδα ελέγχου			E		E		E		E		E
Χειροκίνητο κιβώτιο ταχυτήτων (Για διαρροές)		E	E	E	E	E	E	E	E	E	E
Τιμόνι και σύνδεσμος, εξαρτήματα άξονα & ανάρτησης, άξονας έλικα & μπροστινοί άξονες κίνησης ★			E		E		E		E		E
Τακάκια, ρότορες & άλλα εξαρτήματα φρένων ★		E	E	E	E	E	E	E	E	E	E
Επένδυση φρένων, ρότορες & άλλα εξαρτήματα φρένων ★		E	E	E	E	E	E	E	E	E	E
Ζώνες ασφαλείας, πόρτες, συσπειρωτήρες, αγκύρια & ρυθμιστές		E	E	E	E	E	E	E	E	E	E
Φρένο ποδιών, χειρόφρενο και συμπλέκτης (για ελεύθερη κίνηση, κίνηση και λειτουργία)		E	E	E	E	E	E	E	E	E	E

Φίλτρο κλιματιστικού ★			A		A		A		A		A

Πίνακας 4.2 Διαστήματα συντήρησης του Nissan Micra με κινητήρες βενζίνης.

Συντομογραφίες: Α = Αντικατάσταση, Ε = Επιθεώρηση και διόρθωση ή αντικατάσταση

★ Σε πιο απαιτητικές συνθήκες χρήσης του οχήματος γίνονται πιο σύντομα αυτές οι εργασίες.

ΣΗΜΕΙΩΣΕΙΣ:

(1) Δεν απαιτείται περιοδική συντήρηση. Ωστόσο, εάν αυξηθεί ο θόρυβος της βαλβίδας, ελέγξτε το διάκενο της βαλβίδας.

(2) Αντικαταστήστε τον ιμάντα κίνησης εάν έρθει σε επαφή με καύσιμο ή εντοπιστεί ζημιά κατά την επιθεώρηση.

(3) Χρησιμοποιήστε γνήσιο ψυκτικό κινητήρα NISSAN (μπλε) ή ισοδύναμο στην ποιότητά του, για να αποφύγετε πιθανή διάβρωση αλουμινίου στο σύστημα ψύξης κινητήρα που προκαλείται από τη χρήση μη γνήσιου ψυκτικού κινητήρα.

Ελέγξτε και διορθώστε την αναλογία μείγματος ψυκτικού κινητήρα κάθε 30.000km (18.000 μίλια) ή 24 μήνες. Το πρώτο διάστημα αντικατάστασης είναι 150.000 χιλιόμετρα (90.000 μίλια) ή 96 μήνες. Μετά την πρώτη αντικατάσταση, αντικαταστήστε κάθε 75.000 χιλιόμετρα (54.000 μίλια) ή 48 μήνες. (4) Το φίλτρο καυσίμου δεν χρειάζεται συντήρηση.

Εικόνα 4.4 Nissan Leaf τελευταίας γενιάς.

Ο παρακάτω πίνακας αφορά το Nissan Leaf τελευταίας γενιάς με ηλεκτρικούς κινητήρες (του 2018 μέχρι σήμερα):

Εργασίες συντήρησης		**Διαστήματα συντήρησης**									
Εκτέλεση σε διαστήματα χιλιομέτρων (μίλια) ή μήνα, όποιο από τα δύο συμβεί πρώτο.	km x 1 000 (μίλια x1000) Μήνες	15 (9) 12	30 (18) 24	45 (27) 36	60 (36) 48	75 (45) 60	90 (54) 72	105 (63) 84	120 (72) 96	135 (81) 108	150 (90) 120
Θύρα φόρτισης		E	E	E	E	E	E	E	E	E	E
Λαστιχένιο καπάκι θύρας φόρτισης					E		E		E		E
Σύστημα ψύξης					E		E		E		E
Ψυκτικό συστήματος ψύξης	Σημείωση (1)										
Αναφορά χρήσης μπαταρίας EV		E		E	E	E	E	E	E	E	E
Διάγνωση υπολογιστή		E		E	E	E	E	E	E	E	E
Σύστημα φρένων και υγρό (Για στάθμη και διαρροές)		E	E	E	E	E	E	E	E	E	E
Υγρό φρένων★			A		A		A		A		A
Μειωτικό λάδι κιβωτίου ταχυτήτων		E	E	E	E	E	E	E	E	E	E
Εξοπλισμός διεύθυνσης & σύνδεσμος★			E		E		E		E		E
Ανταλλακτικά άξονα και ανάρτησης★			E		E		E		E		E
Μπροστινοί άξονες κίνησης★			E		E		E		E		E
Τακάκια φρένων, ρότορες & άλλα εξαρτήματα φρένων★		E	E	E	E	E	E	E	E	E	E
Ποδόφρενο, χειρόφρενο (για ελεύθερη κίνηση, χτύπημα και λειτουργία)		E	E	E	E	E	E	E	E	E	E
Φίλτρο κλιματιστικού★			A		A		A		A		R

Πίνακας 4.3 Διαστήματα συντήρησης του Nissan Leaf.

Συντομογραφίες: A = Αντικατάσταση, E = Επιθεώρηση και διόρθωση ή αντικατάσταση

★ Σε πιο απαιτητικές συνθήκες χρήσης του οχήματος γίνονται πιο σύντομα αυτές οι εργασίες.

ΣΗΜΕΙΩΣΕΙΣ:

1) Κατά την προσθήκη ή την αντικατάσταση ψυκτικού υγρού, βεβαιωθείτε ότι χρησιμοποιείτε μόνο Γνήσιο ψυκτικό Nissan Long Life (Μπλε) ή ισοδύναμο με το κατάλληλο μείγμα, για να αποφύγετε πιθανή διάβρωση αλουμινίου στο σύστημα ψύξης που προκαλείται από τη χρήση μη γνήσιου ψυκτικού. Το συνιστώμενο διάστημα συντήρησης του ψυκτικού υγρού πλήρωσης εργοστασίου είναι 200 000 km (125 000 μίλια) ή 15 χρόνια, όποιο από τα δύο συμβεί πρώτο. Η επακόλουθη αντικατάσταση του γνήσιου ψυκτικού υγρού Nissan Long Life (Μπλε) θα πρέπει να γίνεται κάθε 80 000 km (48 000 μίλια) ή 4 χρόνια, όποιο από τα δύο συμβεί πρώτο. Η ανάμειξη οποιουδήποτε άλλου τύπου ψυκτικού ή η χρήση μη απεσταγμένου νερού θα μειώσει το συνιστώμενο διάστημα συντήρησης του ψυκτικού υγρού.

Εικόνα 4.5 Fiat Panda τελευταίας γενιάς

Ο παρακάτω πίνακας αφορά το Fiat Panda τελευταίας γενιάς με κινητήρες βενζίνης:

Οι υπηρεσίες πρέπει να εκτελούνται κάθε 30.000 χλμ. Ή 24 μήνες.

Χιλιάδες μίλια	**18**	**36**	**54**	**72**	**90**	**108**
Χιλιάδες χιλιόμετρα	**30**	**60**	**90**	**120**	**150**	**180**
Μήνες	**24**	**48**	**72**	**96**	**120**	**144**
Ελέγξτε την κατάσταση φόρτισης της μπαταρίας και επαναφορτίστε εάν είναι απαραίτητο	X	X	X	X	X	X
Ελέγξτε την κατάσταση / φθορά των ελαστικών και ρυθμίστε την πίεση, εάν είναι απαραίτητο	X	X	X	X	X	X
Ελέγξτε τη λειτουργία του συστήματος φωτισμού (προβολείς, δείκτες κατεύθυνσης, προειδοποιητικά φώτα κινδύνου, χώρος αποσκευών, θάλαμος επιβατών, κουτί γαντίων, προειδοποιητικά φώτα ταμπλό κ.λπ.)	X	X	X	X	X	X
Ελέγξτε τη λειτουργία του συστήματος πλύσης παρμπρίζ και ρυθμίστε τις πίδακες, εάν είναι απαραίτητο.	X	X	X	X	X	X
Ελέγξτε τη θέση / φθορά του υαλοκαθαριστήρα παρμπρίζ / πίσω παρμπρίζ	X	X	X	X	X	X
Ελέγξτε την καθαριότητα των κλειδαριών καπό και της πίσω πόρτας, καθώς και την	X	X	X	X	X	X

καθαριότητα και τη λίπανση των συνδέσμων						
Επιθεωρήστε οπτικά την κατάσταση: εξωτερικό αμάξωμα, προστασία αμαξώματος, σωλήνες και εύκαμπτους σωλήνες (σύστημα εξάτμισης - καυσίμου - φρένα), λαστιχένια στοιχεία (μπότες, μανίκια, μπιλιοφόροι κ.λπ.)	X	X	X	X	X	X
Ελέγξτε την κατάσταση και τη φθορά του μπροστινού δισκόφρενού	X	X	X	X	X	X
Ελέγξτε την κατάσταση και τη φθορά της επένδυσης του πίσω ταμπούρου		X		X		X
Ελέγξτε και, εάν είναι απαραίτητο, συμπληρώστε τα επίπεδα υγρού (ψύξη κινητήρα, υδραυλικός συμπλέκτης / φρένα, υγρό παρμπρίζ, μπαταρία κ.λπ.)	X	X	X	X	X	X
Επιθεωρήστε οπτικά τις συνθήκες του ιμάντα κίνησης αξεσουάρ		X				X
Ελέγξτε την ένταση του ιμάντα κίνησης αξεσουάρ (εκδόσεις χωρίς αυτόματο τεντωτήρα) (ή κάθε 24 μήνες)	X				X	
Ελέγξτε την κατάσταση του οδοντωτού ιμάντα χρονισμού κίνησης (εξαιρούνται 0,9 TwinAir 85 HP και 0,9 TwinAir Turbo 65 HP κινητήρας)		X				X
Ελέγξτε τη διαδρομή του μοχλού χειρόφρενου και προσαρμόστε, εάν απαιτείται	X	X	X	X	X	X
Ελέγξτε τις εκπομπές καυσαερίων / καπνού	X	X	X	X	X	X
Ελέγξτε τη λειτουργία των συστημάτων διαχείρισης κινητήρα (χρησιμοποιώντας υποδοχή διάγνωσης)	X	X	X	X	X	X
Ελέγξτε τη στάθμη λαδιού του κιβωτίου ταχυτήτων			X			X
Ελέγξτε το διάκενο και ρυθμίστε όπως απαιτείται (έκδοση 1.2 8V 69 HP)		X		X		X
Αντικαταστήστε τα μπουζί (1.2 8V 69 HP και 0.9 TwinAir Turbo 85 HP εκδόσεις) (*)	X	X	X	X	X	X
Αντικαταστήστε τα μπουζί (έκδοση 0,9 TwinAir 65 HP) (*)		X		X		X
Αντικαταστήστε τον (τους) ιμάντα (α)				X		
Αντικαταστήστε τον οδοντωτό ιμάντα χρονισμού (δεν περιλαμβάνονται 0,9 TwinAir 85 HP και 0,9 TwinAir 65 HP κινητήρας) (**)				X		
Αντικατάσταση του φυσιγγίου φίλτρου αέρα (εκδόσεις 1.2 8V 69 HP)		X		X		X

Αντικατάσταση του φυσιγγίου φίλτρου αέρα (0.9 TwinAir Turbo 85 HP και 0.9 εκδόσεις TwinAir 65 HP)	X	X	X	X	X	X
Αλλαγή λαδιού κινητήρα και αντικατάσταση φίλτρου λαδιού (***)	X	X	X	X	X	X
Αλλαγή υγρού φρένων (ή κάθε 24 μήνες)		X		X		X
Αλλάξτε το φίλτρο γύρης (ή κάθε 12 μήνες)	X	X	X	X	X	X

Πίνακας 4.4 Διαστήματα συντήρησης του Fiat Panda με κινητήρες βενζίνης.

ΣΗΜΕΙΩΣΕΙΣ:

(*) Στις εκδόσεις TwinAir 0.9 τα ακόλουθα είναι ζωτικής σημασίας για να διασφαλιστεί η σωστή λειτουργία και να αποφευχθεί σοβαρή ζημιά στον κινητήρα:

(*) Χρησιμοποιείτε μπουζί μόνο πιστοποιημένα ειδικά για αυτούς τους κινητήρες. όλα τα μπουζί πρέπει να είναι του ίδιου τύπου και μάρκας (βλ. παράγραφο "Κινητήρας" στο κεφάλαιο "Τεχνικές προδιαγραφές").

(*) Τηρείτε αυστηρά τα διαστήματα αντικατάστασης των μπουζί που αναφέρονται λεπτομερώς στο προγραμματισμένο πρόγραμμα συντήρησης

(*) Συνιστάται να επικοινωνήσετε με μια αντιπροσωπεία Fiat για αντικατάσταση βύσματος.

(**) Ανεξάρτητα από την απόσταση που καλύπτεται, ο ιμάντας χρονισμού πρέπει να αλλάζει κάθε τέσσερα χρόνια για ιδιαίτερα απαιτητική χρήση (κρύα κλίματα, οδήγηση στην πόλη, μεγάλες περιόδους ρελαντί) ή τουλάχιστον κάθε πέντε χρόνια.

(***) Εάν το όχημα έχει ετήσια χιλιομετρική απόσταση μικρότερη των 10.000 km, το λάδι κινητήρα και το φίλτρο λαδιού κινητήρα πρέπει να αλλάζονται κάθε 12 μήνες.

Ο παρακάτω πίνακας αφορά το Fiat Panda τελευταίας γενιάς με κινητήρες πετρελαίου:

Οι επισκευές πρέπει να εκτελούνται κάθε 35.000 χλμ. Ή 24 μήνες.

Χιλιάδες μίλια	**21**	**42**	**63**	**84**	**105**
Χιλιάδες χιλιόμετρα	**35**	**70**	**105**	**140**	**175**
Μήνες	**24**	**48**	**72**	**96**	**120**
Ελέγξτε την κατάσταση φόρτισης της μπαταρίας και επαναφορτίστε εάν είναι απαραίτητο	X	X	X	X	X
Ελέγξτε την κατάσταση / φθορά των ελαστικών και ρυθμίστε την πίεση, εάν είναι απαραίτητο	X	X	X	X	X
Ελέγξτε τη λειτουργία του συστήματος φωτισμού (προβολείς, δείκτες	X	X	X	X	X

κατεύθυνσης, προειδοποιητικά φώτα κινδύνου, χώρος αποσκευών, θάλαμος επιβατών, κουτί γαντίων, προειδοποιητικά φώτα ταμπλό κ.λπ.)					
Ελέγξτε τη λειτουργία του συστήματος πλύσης παρμπρίζ και ρυθμίστε τις πίδακες, εάν είναι απαραίτητο.	X	X	X	X	X
Ελέγξτε τη θέση / φθορά του υαλοκαθαριστήρα παρμπρίζ / πίσω παρμπρίζ	X	X	X	X	X
Ελέγξτε την καθαριότητα των κλειδαριών καπό και της πίσω πόρτας, καθώς και την καθαριότητα και τη λίπανση των συνδέσμων	X	X	X	X	X
Επιθεωρήστε οπτικά την κατάσταση: εξωτερικό αμάξωμα, προστασία αμαξώματος, σωλήνες και εύκαμπτους σωλήνες (σύστημα εξάτμισης - καυσίμου - φρένα), λαστιχένια στοιχεία (μπότες, μανίκια, μπιλιοφόροι κ.λπ.)	X	X	X	X	X
Ελέγξτε την κατάσταση και τη φθορά του μπροστινού δισκόφρενού	X	X	X	X	X
Ελέγξτε την κατάσταση και τη φθορά της επένδυσης του πίσω ταμπούρου		X		X	
Ελέγξτε και, εάν είναι απαραίτητο, συμπληρώστε τα επίπεδα υγρού (ψύξη κινητήρα, υδραυλικός συμπλέκτης / φρένα, υγρό παρμπρίζ, μπαταρία κ.λπ.)	X	X	X	X	X
Επιθεωρήστε οπτικά τις συνθήκες του ιμάντα κίνησης		X			X
Ελέγξτε τη διαδρομή του μοχλού χειρόφρενου και προσαρμόστε, εάν απαιτείται	X	X	X	X	X
Ελέγξτε τις εκπομπές καυσαερίων / καπνού	X	X	X	X	X
Ελέγξτε τη λειτουργία των συστημάτων διαχείρισης κινητήρα (χρησιμοποιώντας υποδοχή διάγνωσης)	X	X	X	X	X
Ελέγξτε τη στάθμη λαδιού του κιβωτίου ταχυτήτων	X	X	X	X	X
Αντικατάσταση του φυσιγγίου φίλτρου καυσίμου		X		X	
Αντικαταστήστε τον (τους) ιμάντα (α)			X		
Αντικατάσταση του φυσιγγίου φίλτρου αέρα	X	X	X	X	X
Αλλαγή λαδιού κινητήρα και αντικατάσταση φίλτρου λαδιού (ή κάθε 24 μήνες) (*)(**)					
Αλλαγή υγρού φρένων (ή κάθε 24 μήνες)		X		X	

Αλλάξτε το φίλτρο γύρης (ή κάθε 12 μήνες)	X	X	X	X	X

Πίνακας 4.5 Διαστήματα συντήρησης του Fiat Panda με κινητήρες βενζίνης.

ΣΗΜΕΙΩΣΕΙΣ:

(*) Εάν το όχημα χρησιμοποιείται κυρίως για οδήγηση στην πόλη ή έχει ετήσια χιλιομετρική απόσταση μικρότερη από 10.000 χλμ., Το λάδι κινητήρα και φίλτρο λαδιού αλλάζει κάθε 12 μήνες.

(**) Το πραγματικό διάστημα αντικατάστασης φίλτρου λαδιού και λαδιού κινητήρα εξαρτάται από τις συνθήκες χρήσης του αυτοκινήτου και υποδεικνύεται από ένα μήνυμα προειδοποιητικής λυχνίας στο ταμπλό οργάνων ή σε κάθε περίπτωση κάθε 24 μήνες.

Εικόνα 4.6 Fiat Tipo τελευταίας γενιάς

Ο παρακάτω πίνακας αφορά το Fiat Tipo τελευταίας γενιάς με κινητήρες πετρελαίου:

Χιλιάδες μίλια	**12**	**24**	**36**	**48**	**60**	**72**	**84**	**96**	**108**	**120**
Χιλιάδες χιλιόμετρα	**20**	**40**	**60**	**80**	**100**	**120**	**140**	**160**	**180**	**200**
Μήνες	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Ελέγξτε την κατάσταση/φθορά των ελαστικών και ρυθμίστε την πίεση, εάν χρειάζεται. Ελέγξτε τις συνθήκες επαναφόρτισης/ημερομηνία λήξης του κιτ γρήγορης επισκευής ελαστικών (όπου παρέχεται)	X	X	X	X	X	X	X	X	X	X
Ελέγξτε τη λειτουργία του συστήματος φωτισμού (προβολείς, φλας,	X	X	X	X	X	X	X	X	X	X

προειδοποιητικά φώτα κινδύνου, χώρο αποσκευών, χώρο επιβατών, ντουλαπάκι γαντιών, προειδοποιητικά φώτα ταμπλό, κ.λπ.)										
Ελέγξτε και συμπληρώστε τη στάθμη του υγρού, εάν απαιτείται, στο χώρο του κινητήρα (1) (2)	X	X	X	X	X	X	X	X	X	X
Ελέγξτε τις εκπομπές καυσαερίων/καπνό	X	X	X	X	X	X	X	X	X	X
Χρησιμοποιήστε την υποδοχή διάγνωσης για να ελέγξετε τη λειτουργία συστήματος ελέγχου κινητήρα, εκπομπών και φθοράς λαδιών κινητήρα (το τελευταίο, όπου προβλέπεται) (3)	X	X	X	X	X	X	X	X	X	X
Επιθεωρήστε οπτικά την κατάσταση: εξωτερικό αμάξωμα, προστασία κάτω από το αμάξωμα, σωλήνες και εύκαμπτοι σωλήνες (εξάτμιση, σύστημα καυσίμου, φρένα), ελαστικά στοιχεία (γκέτες, μανίκια, δακτύλιοι κ.λπ.)		X		X		X		X		X
Ελέγξτε τη θέση/φθορά του υαλοκαθαριστήρα και των λεπίδων του υαλοκαθαριστήρα του πίσω παρμπρίζ (όπου παρέχεται)		X		X		X		X		X
Ελέγξτε τη λειτουργία του συστήματος υαλοκαθαριστήρα/πλυστήρα παρμπρίζ και ρυθμίστε τα ακροφύσια, εάν χρειάζεται		X		X		X		X		X
Ελέγξτε την καθαριότητα της κουκούλας και των κλειδαριών του χώρου αποσκευών, την καθαριότητα και τη λίπανση των συνδέσμων		X		X		X		X		X
Ελέγξτε τη διαδρομή του μοχλού του χειρόφρενου και ρυθμίστε, εάν χρειάζεται		X		X		X		X		X
Επιθεωρήστε οπτικά τις συνθήκες και τη φθορά των μπροστινών και πίσω τακακιών δισκόφρενου (4) και την ακεραιότητα του δείκτη φθοράς των τακακιών	X	X	X	X	X	X	X	X	X	X
Επιθεωρήστε οπτικά την κατάσταση και τη φθορά της επένδυσης του πίσω τυμπάνου φρένου (4)	X	X	X	X	X	X	X	X	X	X
Επιθεωρήστε οπτικά την κατάσταση του οδοντωτού ιμάντα χρονισμού (1.6 Multijet 130HP E6D-τελικές εκδόσεις) (5)			X			X			X	
Επιθεωρήστε οπτικά την κατάσταση των ιμάντων μετάδοσης κίνησης αξεσουάρ (5)			X			X			X	
Ελέγξτε την τάνυση του ιμάντα μετάδοσης κίνησης αξεσουάρ (εκδόσεις χωρίς αυτόματο τεντωτήρα)			X			X			X	
Ελέγξτε και, εάν χρειάζεται, συμπληρώστε τη στάθμη λαδιού του ηλεκτροϋδραυλικού ενεργοποιητή (6)						X				
Αλλάξτε λάδι κινητήρα και αντικαταστήστε το φίλτρο λαδιού (εκδόσεις με DPF) (7)										
Αντικαταστήστε τον/τους ιμάντες μετάδοσης κίνησης αξεσουάρ (5)										

Αντικατάσταση φυσιγγίου φίλτρου καυσίμου (1.3 Multijet 95HP E6D-τελικές εκδόσεις) (8)			X			X			X	
Αντικατάσταση φυσιγγίου φίλτρου καυσίμου (1.6 Multijet 130HP E6D-τελικές εκδόσεις) (8)		X		X		X		X		X
Αντικαταστήστε την κασέτα φίλτρου αέρα (9)		X		X		X		X		X
Αλλάξτε το υγρό φρένων (10)										
Αντικαταστήστε το καθαριστικό χώρου επιβατών (9)		X		X		X		X		X

Πίνακας 4.6 Διαστήματα συντήρησης του Fiat Tipo με κινητήρες πετρελαίου.

ΣΗΜΕΙΩΣΕΙΣ:

(1) Χρησιμοποιείτε πάντα τα υγρά που αναφέρονται στο εγχειρίδιο για συμπλήρωση και μόνο αφού ελέγξετε ότι το σύστημα είναι άθικτο.

(2) Η κατανάλωση πρόσθετου εκπομπών (UREA) εξαρτάται από την κατάσταση χρήσης του οχήματος και υποδεικνύεται με προειδοποιητική λυχνία και μήνυμα στον πίνακα οργάνων (για εκδόσεις/αγορές, όπου παρέχεται).

(3) Εάν το "εναπομένον ποσοστό αποδοτικού λαδιού κινητήρα" που ανιχνεύεται από τα διαγνωστικά του οχήματος είναι χαμηλότερο ή ίσο με 20%, συνιστάται η αντικατάσταση του λαδιού κινητήρα και του φίλτρου κινητήρα για να αποφευχθεί άλλη λειτουργία σέρβις μετά από σύντομο χρονικό διάστημα.

(4) Για εκδόσεις/αγορές, όπου παρέχονται

(5) Η μέγιστη χιλιομετρική απόσταση είναι 120.000 km. Ο ιμάντας πρέπει να αντικαθίσταται κάθε 6 χρόνια, ανεξάρτητα από την απόσταση που διανύθηκε. Εάν το όχημα χρησιμοποιείται σε βαριές συνθήκες (περιοχές με σκόνη, ιδιαίτερα δύσκολες καιρικές συνθήκες, πολύ χαμηλές ή πολύ υψηλές θερμοκρασίες για μεγάλα χρονικά διαστήματα, οδήγηση στην πόλη, μεγάλες περιόδους ρελαντί), η μέγιστη χιλιομετρική απόσταση είναι 60.000 km. Ο ιμάντας πρέπει να αντικαθίσταται κάθε 4 χρόνια ανεξάρτητα από τα χιλιόμετρα.

(6) Έλεγχος που πρέπει να διενεργείται κάθε χρόνο για οχήματα στο δρόμο σε χώρες με ιδιαίτερα βαριά κλίματα (ψυχρές χώρες).

(7) Το διάστημα αλλαγής λαδιού κινητήρα και φίλτρου εξαρτάται από τις συνθήκες οδήγησης και σηματοδοτείται από μια προειδοποιητική λυχνία ή μήνυμα στον πίνακα οργάνων. Σε κάθε περίπτωση, δεν πρέπει ποτέ να υπερβαίνει τα 2 χρόνια. Όπου το αυτοκίνητο χρησιμοποιείται κυρίως σε αστικές περιοχές, πρέπει να αντικαθιστάτε το φίλτρο λαδιού κινητήρα κάθε χρόνο.

(8) Εάν το αυτοκίνητο λειτουργεί με καύσιμο με ποιότητα κάτω από τις σχετικές ευρωπαϊκές προδιαγραφές, αυτό το φίλτρο πρέπει να αντικαθίσταται κάθε 20.000 km

(9) Εάν το όχημα χρησιμοποιείται σε περιοχές με σκόνη, αυτό το καθαριστικό θα πρέπει να αντικαθίσταται κάθε 20.000 km.

(10) Η αντικατάσταση του υγρού φρένων πρέπει να γίνεται κάθε δύο χρόνια, ανεξάρτητα από τα χιλιόμετρα.

Ο παρακάτω πίνακας αφορά το Fiat Tipo τελευταίας γενιάς με κινητήρες βενζίνης:

Χιλιάδες μίλια	**9**	**18**	**27**	**36**	**45**	**54**	**63**	**72**	**81**	**90**
Χιλιάδες χιλιόμετρα	**15**	**30**	**45**	**60**	**75**	**90**	**105**	**120**	**135**	**150**
Μήνες	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Ελέγξτε την κατάσταση/φθορά των ελαστικών και ρυθμίστε την πίεση, εάν χρειάζεται. Ελέγξτε τις συνθήκες επαναφόρτισης/ημερομηνία λήξης του κιτ γρήγορης επισκευής ελαστικών (όπου παρέχεται)	X	X	X	X	X	X	X	X	X	X
Ελέγξτε τη λειτουργία του συστήματος φωτισμού (προβολείς, φλας, προειδοποιητικά φώτα κινδύνου, χώρο αποσκευών, χώρο επιβατών, ντουλαπάκι γαντιών, προειδοποιητικά φώτα ταμπλό, κ.λπ.)	X	X	X	X	X	X	X	X	X	X
Ελέγξτε και συμπληρώστε τη στάθμη του υγρού, εάν απαιτείται, στο χώρο του κινητήρα (1)	X	X	X	X	X	X	X	X	X	X
Ελέγξτε τις εκπομπές καυσαερίων/καπνό	X	X	X	X	X	X	X	X	X	X
Χρησιμοποιήστε την υποδοχή διάγνωσης για να ελέγξετε τη λειτουργία συστήματος ελέγχου κινητήρα, εκπομπών και φθοράς λαδιών κινητήρα (το τελευταίο, όπου προβλέπεται) (3)	X	X	X	X	X	X	X	X	X	X
Επιθεωρήστε οπτικά την κατάσταση: εξωτερικό αμάξωμα, προστασία κάτω από το αμάξωμα, σωλήνες και εύκαμπτοι σωλήνες (εξάτμιση, σύστημα καυσίμου, φρένα), ελαστικά στοιχεία (γκέτες, μανίκια, δακτύλιοι κ.λπ.)		X		X		X		X		X
Ελέγξτε τη θέση/φθορά του υαλοκαθαριστήρα και των λεπίδων του υαλοκαθαριστήρα του πίσω παρμπρίζ (όπου παρέχεται)		X		X		X		X		X
Ελέγξτε τη λειτουργία του συστήματος υαλοκαθαριστήρα/πλυστήρα παρμπρίζ και ρυθμίστε τα ακροφύσια, εάν χρειάζεται		X		X		X		X		X
Ελέγξτε την καθαριότητα της κουκούλας και των κλειδαριών του χώρου αποσκευών, την καθαριότητα και τη λίπανση των συνδέσμων		X		X		X		X		X
Ελέγξτε τη διαδρομή του μοχλού του χειρόφρενου και ρυθμίστε, εάν χρειάζεται		X		X		X		X		X
Επιθεωρήστε οπτικά τις συνθήκες και τη φθορά των μπροστινών και πίσω τακακιών δισκόφρενου και την ακεραιότητα του δείκτη φθοράς των τακακιών	X	X	X	X	X	X	X	X	X	X
Επιθεωρήστε οπτικά την κατάσταση και τη φθορά της επένδυσης του πίσω τυμπάνου φρένου (3)	X	X	X	X	X	X	X	X	X	X

Επιθεωρήστε οπτικά την κατάσταση του οδοντωτού ιμάντα χρονισμού (4)			X			X			X	
Επιθεωρήστε οπτικά την κατάσταση των ιμάντων μετάδοσης κίνησης αξεσουάρ			X			X			X	
Ελέγξτε την τάνυση του ιμάντα μετάδοσης κίνησης αξεσουάρ (εκδόσεις χωρίς αυτόματο τεντωτήρα)			X			X			X	
Ελέγξτε και, εάν χρειάζεται, συμπληρώστε τη στάθμη λαδιού του ηλεκτροϋδραυλικού ενεργοποιητή						X				
Αλλάξτε λάδι κινητήρα και αντικαταστήστε το φίλτρο λαδιού (5)		X		X		X		X		X
Αντικατάσταση μπουζί				X				X		
Αντικαταστήστε τον οδοντωτό ιμάντα χρονισμού (4)										
Αντικαταστήστε τον/τους ιμάντες μετάδοσης κίνησης αξεσουάρ (4)										
Αντικαταστήστε την κασέτα φίλτρου αέρα (8)		X		X		X		X		X
Αλλάξτε το υγρό φρένων (10)										
Αντικαταστήστε το καθαριστικό χώρου επιβατών (9)		X		X		X		X		X

Πίνακας 4.7 Διαστήματα συντήρησης του Fiat Tipo με κινητήρες βενζίνης.

ΣΗΜΕΙΩΣΕΙΣ:

(1) Χρησιμοποιείτε πάντα τα υγρά που αναφέρονται στο εγχειρίδιο για συμπλήρωση και μόνο αφού ελέγξετε ότι το σύστημα είναι άθικτο.

(2) Εάν η ετήσια χιλιομετρική απόσταση του αυτοκινήτου είναι μικρότερη από 10.000 km, το λάδι κινητήρα και το φίλτρο πρέπει να αντικαθίστανται κάθε χρόνο.

(3) Εάν η ποιότητα του λαδιού κινητήρα που ανιχνεύεται από το διαγνωστικό του οχήματος είναι χαμηλότερη από 20%, συνιστάται η αντικατάσταση του λαδιού κινητήρα και του φίλτρου κινητήρα για να αποφευχθεί άλλη εργασία συντήρησης μετά από σύντομο χρονικό διάστημα.

(4) Η μέγιστη χιλιομετρική απόσταση είναι 120.000 km. Ο ιμάντας πρέπει να αντικαθίσταται κάθε 6 χρόνια, ανεξάρτητα από την απόσταση που διανύθηκε. Εάν το όχημα χρησιμοποιείται σε βαριές συνθήκες (περιοχές με σκόνη, ιδιαίτερα δύσκολες καιρικές συνθήκες, πολύ χαμηλές ή πολύ υψηλές θερμοκρασίες για μεγάλα χρονικά διαστήματα, οδήγηση στην πόλη, μεγάλες περιόδους ρελαντί), η μέγιστη χιλιομετρική απόσταση είναι 60.000 km. Ο ιμάντας πρέπει να αντικαθίσταται κάθε 4 χρόνια ανεξάρτητα από τα χιλιόμετρα.

(5) Το διάστημα αλλαγής λαδιού κινητήρα και φίλτρου εξαρτάται από τις συνθήκες οδήγησης και σηματοδοτείται από μια προειδοποιητική λυχνία ή ένα μήνυμα στον πίνακα οργάνων. Σε καμία περίπτωση, μην υπερβαίνετε ποτέ το 1 έτος.

(6) Για να διασφαλιστεί η σωστή λειτουργία και να αποτραπεί σοβαρή ζημιά στον κινητήρα, είναι απαραίτητο να ενεργήσετε ως εξής: χρησιμοποιείτε μόνο μπουζί ειδικά πιστοποιημένα για αυτούς τους κινητήρες. όλα τα μπουζί πρέπει να είναι του ίδιου τύπου και μάρκας (δείτε την παράγραφο "Κινητήρας" στο κεφάλαιο "Τεχνικές προδιαγραφές"). τηρείτε αυστηρά τα

διαστήματα αντικατάστασης μπουζί στο Πρόγραμμα σέρβις. Συνιστάται να επικοινωνήσετε με την αντιπροσωπεία της FIAT για αντικατάσταση βύσματος.

(8) Εάν το όχημα χρησιμοποιείται σε περιοχές με σκόνη, αυτό το φίλτρο θα πρέπει να αντικαθίσταται κάθε 15.000 km.

(10) Η αντικατάσταση του υγρού φρένων πρέπει να γίνεται κάθε δύο χρόνια, ανεξάρτητα από τα χιλιόμετρα.

Εικόνα 4.7 Smart Fortwo τελευταίας γενιάς.

Ο παρακάτω πίνακας αφορά το Smart Fortwo τελευταίας γενιάς με κινητήρες βενζίνης:

	Service A 15,000 KM	**Service B 30,000 KM**	**Service A 45,000 KM**	**Service B 60,000 KM**	**Service A 75,000 KM**	**Service B 90,000 KM**
Συνθετικό λάδι κινητήρα & αλλαγή φίλτρου	X	X	X	X	X	X
Αλλαγή φίλτρου συνδυασμού	X	X	X	X	X	X
Αντικατάσταση φίλτρου αέρα		^	[]	^		X
αντικατάσταση μπαταρίας smartKey		X		X		X
Αντικατάσταση μπουζί			X			X
Αντικατάσταση ιμάντων V						[]

Αλλαγή υγρού φρένων	[] Κάθε δύο χρόνια / ^ κάθε τρία χρόνια					
Αναπλήρωση υγρού πλυσίματος	X	X	X	X	X	X
Αντικατάσταση ψυκτικού				[]		^
Έλεγχος και διόρθωση επιπέδων υγρών	X	X	X	X	X	X
Έλεγχος και διόρθωση πίεσης ελαστικών	X	X	X	X	X	X
Επιθεώρηση τακακιών φρένων (εμπρός)	X	X	X	X	X	X
Επιθεώρηση τακακιών φρένων (πίσω)				^		X
Ρύθμιση βαλβίδας						[]
Επαναφορά μετρητή συντήρησης	X	X	X	X	X	X

Πίνακας 4.8 Διαστήματα συντήρησης του Smart Fortwo με κινητήρες βενζίνης.

^ Smart 453 (έτος μοντέλου 2016 και νεότερο) μόνο. Μόνο [] Smart 451 (έτος μοντέλου 2015 και προηγούμενα).

Ο παρακάτω πίνακας αφορά το Smart Fortwo τελευταίας γενιάς με ηλεκτρικό κινητήρα:

	Service A 20,000 KM	**Service B 40,000 KM**	**Service A 60,000 KM**	**Service B 80,000 KM**
Αλλαγή φίλτρου συνδυασμού	X	X	X	X
αντικατάσταση μπαταρίας smartKey	X	X	X	X
Αλλαγή υγρού φρένων	Κάθε δύο χρόνια			
Αντικατάσταση Αποξηραντικού φυσιγγιού	Κάθε δύο χρόνια			
Αναπλήρωση υγρού πλυσίματος	X	X	X	X
Ελέγξτε την ηλεκτρική μονάδα και το κιβώτιο ταχυτήτων για ζημιές, διαρροές και ελαττωματική τοποθέτηση	X	X	X	X
Ελέγξτε τους συνδετήρες και την κατάσταση	X	X	X	X

όλων των γραμμών υψηλής τάσης				
Έλεγχος και διόρθωση επιπέδων υγρών	X	X	X	X
Ελέγξτε την μπαταρία υψηλής τάσης	X	X	X	X
Έλεγχος και διόρθωση πίεσης ελαστικών	X	X	X	X
Επιθεώρηση τακακιών φρένων (εμπρός)	X	X	X	X
Επιθεώρηση τακακιών φρένων (πίσω)				X
Επαναφορά μετρητή συντήρησης	X	X	X	X

Πίνακας 4.9 Διαστήματα συντήρησης του Smart Fortwo με ηλεκτρικούς κινητήρες.

Με τις παραπάνω πληροφορίες, δηλαδή τους πίνακες και τις απαντήσεις από τις συνεντεύξεις έχει συγκεντρωθεί αρκετό υλικό ώστε να είναι δυνατός ο σχεδιασμός της βάσης δεδομένων της εφαρμογής Carbook. Στα επόμενα υποκεφάλαια ακολουθεί το στάδιο αυτό.

4.2 Αρχιτεκτονική της εφαρμογής CarBook

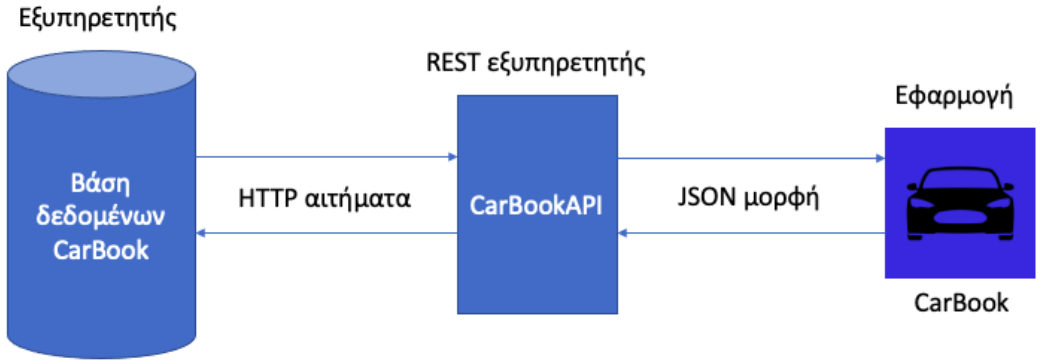
Η αρχιτεκτονική (εικόνα 4.8) η οποία χρησιμοποιήθηκε για την εφαρμογή CarBook ακολουθεί το μοντέλο πελάτη – εξυπηρετητή. Η εφαρμογή συνδέεται στον εξυπηρετητή και αντλεί δεδομένα από την βάση δεδομένων που βρίσκεται στον εξυπηρετητή, η οποία περιέχει γενικές πληροφορίες σχετικά με τις κατασκευάστριες εταιρίες αυτοκινήτων, τα μοντέλα αυτοκινήτων και τα συνεργεία που έχουν καταχωρηθεί από τον προγραμματιστή (το κεφάλαιο 4.1 περιέχει το δείγμα δεδομένων που έχει συλλεχθεί από τις επίσημες αντιπροσωπείες).

Για την διασύνδεση της εφαρμογής για android με τον εξυπηρετητή που περιέχει την βάση δεδομένων, αναπτύχθηκε διαδικτυακή υπηρεσία (αγγλικά: web service) τύπου REST. Η υπηρεσία αυτή καθιστά δυνατή την αποστολή και λήψη αιτημάτων HTTP μεταξύ του εξυπηρετητή και του πελάτη υλοποιώντας τις απαραίτητες μεθόδους για την λήψη των δεδομένων. Δηλαδή στην επικοινωνία του πελάτη – εξυπηρετητή μεσολαβεί ο REST εξυπηρετητής και κάνει αποστολή και λήψη των δεδομένων σε JSON μορφή, την οποία αναγνωρίζει ο πελάτης (εφαρμογή android CarBook).

Για πληροφορίες πιο ειδικές η αποθήκευση γίνεται σε τοπική βάση δεδομένων στην ίδια την εφαρμογή (SQLite βάση δεδομένων). Παράδειγμα αυτού του είδους πληροφορίας αποτελεί το αυτοκίνητο που έχει καταχωρήσει ο χρήστης της εφαρμογής με τις πληροφορίες του (αριθμός πινακίδας, χιλιόμετρα στο οδόμετρο, αριθμός πλαισίου και αλλά) που είναι μοναδικές για το συγκεκριμένο όχημα.

Συνοψίζοντας, η εφαρμογή CarBook αποθηκεύει τοπικά τις ειδικές πληροφορίες που αφορούν τον χρήστη. Για τις πιο γενικές πληροφορίες που αφορούν τα οχήματα συνδέεται

στον REST εξυπηρετητή οποίος στέλνει και λαμβάνει αιτήματα από τον εξυπηρετητή της βάσης δεδομένων και απαντάει με δεδομένα σε JSON μορφή στην εφαρμογή.



Εικόνα 4.8 Διάγραμμα αρχιτεκτονικής του συστήματος CarBook.

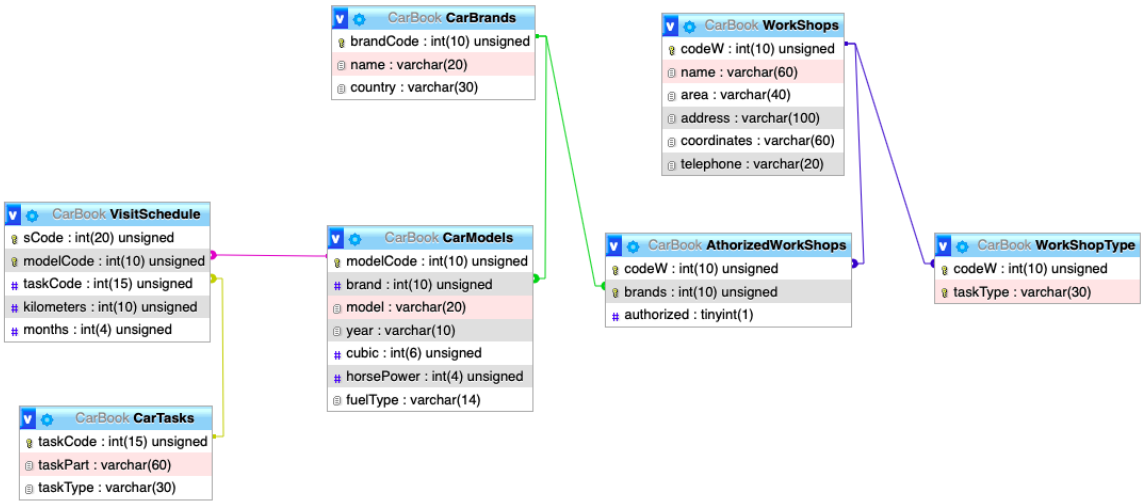
4.3 Σχεδιασμός και υλοποίηση της βάσης δεδομένων CarBook

4.2.1 Βάση δεδομένων MySQL

Η βάση δεδομένων είναι ένα από τα βασικά στοιχεία μιας λειτουργικής εφαρμογής. Χωρίς τη βάση δεδομένων η εφαρμογή θα ήταν δύσκολο να υλοποιηθεί, επειδή δεν θα ήταν εύκολο να ανακτηθούν τα δεδομένα. Με τα δεδομένα που αποκτήθηκαν από την έρευνα είναι πλέον δυνατό να γίνει ο σχεδιασμός της βάσης δεδομένων. Για να γίνει αυτή η διαδικασία χρειάζεται να μελετηθούν οι πληροφορίες και η ροή που υπάρχει όταν ένας ιδιοκτήτης επισκέπτεται ένα συνεργείο.

Αρχικά πρέπει να γνωρίζει ο μηχανικός κάποια βασικά χαρακτηριστικά για το όχημα, για παράδειγμα τι τύπου καυσίμου είναι η μηχανή του αυτοκινήτου, πόσους ίππους δίνει κ.τ.λ. Ταυτόχρονα πρέπει να έχει στην διάθεση του το ιστορικό των παλαιότερων επισκευών που έχουν γίνει και το επίσημο πρόγραμμα συντήρησης για το μοντέλο αυτοκινήτου του κατόχου από την κατασκευαστική εταιρία. Από την πλευρά του κατόχου είναι ορθό να υπάρχει πληροφορία για τα συνεργεία που σχετίζονται με το όχημα του, ώστε στο μέλλον να μπορεί να τα αναζητήσει αν χρειαστεί, και τέλος κάποιες βασικές πληροφορίες για το όχημα που του ανήκει. Έτσι προκύπτουν πληροφορίες που είναι πιο γενικές (όπως τα μοντέλα) και πιο ειδικές (όπως τα στοιχεία ενός συγκεκριμένου οχήματος). Η βάση CarBook περιέχει τις οντότητες οι οποίες είναι γενικές. Στο επόμενο υποκεφάλαιο γίνεται περιγραφή της διαχείρισης των πληροφορίων που είναι πιο ειδικές.

Για να γίνει μοντελοποίηση του συστήματος επιλέχθηκαν οι εξής οντότητες: Κατασκευαστές αυτοκίνητων, Μοντέλα αυτοκινήτων, Βιβλίο επισκευών, Πρόγραμμα επισκευών και Συνεργεία. Αυτές είναι οι κύριες αλλά αναλύοντας τες σε συνδυασμό με τις σχέσεις που έχουν μεταξύ τους είναι αναγκαίο κάποιες να αναλυθούν παραπάνω και να γίνουν περισσότερες. Παρακάτω φαίνεται το διάγραμμα οντοτήτων συσχετίσεων (ERD) της βάσης δεδομένων και μετά ακολουθεί περαιτέρω ανάλυση (εικόνα 4.9).



Εικόνα 4.9 Διάγραμμα οντοτήτων συσχετίσεων (ERD) της βάσης δεδομένων.

Πίνακες:

CarBrands: Το περιεχόμενο του πίνακα αυτού είναι όλες οι μάρκες των αυτοκινήτων που καταχωρούνται ως μοντέλα στην βάση δεδομένων. Αποτελείται από τις στήλες:

- **brandCode:** Κωδικός μάρκας αυτοκινήτων (κύριο κλειδί)
- **name:** Όνομα μάρκας αυτοκινήτων
- **country:** Χώρα στην οποία εδρεύει η εταιρία

CarModels: Το περιεχόμενο του πίνακα αυτού είναι όλα τα μοντέλα των αυτοκινήτων που καταχωρούνται στην βάση δεδομένων και υποστηρίζονται από την εφαρμογή. Αποτελείται από τις στήλες:

- **modelCode:** Κωδικός μοντέλου αυτοκίνητου (κύριο κλειδί)
- **brand:** Κωδικός μάρκας αυτοκινήτου (ξένο κλειδί)
- **model:** Όνομα μοντέλου αυτοκινήτου
- **year:** Χρόνος κατασκευής αυτοκινήτου
- **cubic:** Κυβισμός κινητήρα
- **horsepower:** Ίπποι κινητήρα
- **fuelType:** Τύπος καυσίμου μοντέλου αυτοκινήτου

CarTasks: Το περιεχόμενο του πίνακα αυτού είναι όλες οι εργασίες συντήρησης ή επισκευής που μπορούν να γίνουν σε ένα αυτοκίνητο. Αποτελείται από τις στήλες:

- **taskCode:** Κωδικός εργασίας (κύριο κλειδί)
- **taskPart:** Όνομα εργασίας
- **taskType**: Τύπος εργασίας

VisitSchedule: Το περιεχόμενο του πίνακα αυτού είναι τα διαστήματα όπου γίνεται η συντήρηση του κάθε αυτοκινήτου. Για κάθε μοντέλο καταχωρημένο στην βάσης δεδομένων υπάρχουν οι αντίστοιχες καταχωρήσεις. Αποτελείται από τις στήλες:

- **sCode:** Κωδικός προγραμματισμένης συντήρησης (σύνθετο κύριο κλειδί)

- **modelCode:** Κωδικός μοντέλου αυτοκινήτου (σύνθετο κύριο κλειδί, ξένο κλειδί)
- **taskCode:** Κωδικός εργασίας (ξένο κλειδί)
- **kilometers:** Χιλιόμετρα που πρέπει να περάσουν για να πραγματοποιηθεί μια εργασία
- **months:** Μήνες που πρέπει να περάσουν για να πραγματοποιηθεί μια εργασία

WorkShops: Το περιεχόμενο του πίνακα αυτού είναι τα συνεργεία που είναι καταχωρημένα στην βάση δεδομένων. Αποτελείται από τις στήλες:

- **codeW:** Κωδικός συνεργείου (κύριο κλειδί)
- **name:** Όνομα συνεργείου
- **area:** Περιοχή στην οποία βρίσκεται το συνεργείο
- **address:** Διεύθυνση συνεργείου
- **coordinates:** Συντεταγμένες συνεργείου
- **telephone:** Τηλέφωνο συνεργείου

WorkshopType: Το περιεχόμενο του πίνακα αυτού είναι οι τύποι εργασιών που υποστηρίζει το κάθε συνεργείο. Αποτελείται από τις στήλες:

- **codeW:** Κωδικός συνεργείου (σύνθετο κύριο κλειδί, ξένο κλειδί)
- **taskType**: Τύπος εργασίας (σύνθετο κύριο κλειδί)

AuthorizedWorkShops: Το περιεχόμενο του πίνακα αυτού είναι οι μάρκες αυτοκινήτων που υποστηρίζει το κάθε συνεργείο και αν είναι εξουσιοδοτημένο το συνεργείο. Αποτελείται από τις στήλες:

- **codeW:** Κωδικός συνεργείου (σύνθετο κύριο κλειδί, ξένο κλειδί)
- **brands:** Κωδικός μάρκας αυτοκινήτου (σύνθετο κύριο κλειδί, ξένο κλειδί)
- **authorized:** Δυαδική τιμή για κάθε μάρκα αυτοκινήτου που υποστηρίζεται

4.2.2 Βάση δεδομένων SQLite

Η βάση δεδομένων όπως προαναφέρθηκε είναι ένα από τα βασικά στοιχεία μιας εφαρμογής και σκοπός της είναι να αποθηκεύει αποδοτικά τα δεδομένα. Υπάρχουν δεδομένα που η προσπέλαση τους γίνεται κατά απαίτηση και δεδομένα που η χρήση τους είναι πιο άμεση και παράλληλα έχουν μικρή έκταση.

Για την αποθήκευση των δεδομένων αυτών, δημιουργήθηκε στην εφαρμογή CarBook η βάση δεδομένων myCars η οποία αποθηκεύει τοπικά στην εφαρμογή πληροφορίες για τα οχήματα του χρήστη, για τις εργασίες που έγιναν στα οχήματα και τις υπενθυμίσεις που δημιουργούνται. Η τοπική αυτή βάση σε συνεργασία με την βάση δεδομένων CarBook (σε MySQL) αποτελούν μια ολοκληρωμένη υλοποίηση, ικανή να διαχειριστεί την απαραίτητη πληροφορία. Παρακάτω ακολουθεί η ανάλυση των πινάκων:

Πίνακες:

tbl_cars: Το περιεχόμενο του πίνακα αυτού είναι οι βασικές πληροφορίες για τα οχήματα που έχει καταχωρήσει ο χρήστης στην εφαρμογή. Αποτελείται από τις στήλες:

- **id:** Κωδικός οχήματος του χρήστη (κύριο κλειδί)
- **ModelCode:** Κωδικός μοντέλου αυτοκινήτου

- **DateDay:** Ημέρα αγοράς οχήματος χρήστη
- **DateMonth:** Μήνας αγοράς οχήματος χρήστη
- **DateYear:** Χρόνος αγοράς οχήματος χρήστη
- **Chassis:** Αριθμός πλαισίου οχήματος του χρήστη
- **Plate:** Αριθμός πινακίδας οχήματος του χρήστη
- **Color:** Χρώμα οχήματος του χρήστη

tbl_car_visit: Το περιεχόμενο του πίνακα αυτού είναι οι πληροφορίες που αφορούν την επίσκεψη του χρήστη σε ένα συνεργείο. Αποτελείται από τις στήλες:

- **visit_code:** Κωδικός επίσκεψης σε συνεργείο για ένα όχημα του χρήστη (σύνθετο κύριο κλειδί)
- **mycar_code:** Κωδικός του οχήματος του χρήστη (σύνθετο κύριο κλειδί, ξένο κλειδί (id από tbl_cars)
- **speedometer:** Αριθμός χιλιομέτρων οδομέτρου όταν πραγματοποιήθηκε η επίσκεψη
- **next_service:** Αριθμός χιλιομέτρων που πρέπει να γίνει η επόμενη επίσκεψη στο συνεργείο
- **w_code:** Κωδικός συνεργείου στο οποίο έγινε η επίσκεψη (ξένο κλειδί)
- **p_date:** Ημερομηνία επίσκεψης

tbl_car_service: Το περιεχόμενο του πίνακα αυτού είναι οι εργασίες που πραγματοποιήθηκαν σε κάθε επίσκεψη για το κάθε ένα όχημα του χρήστη. Αποτελείται από τις στήλες:

- **visit_code:** Κωδικός επίσκεψης σε συνεργείο για ένα όχημα του χρήστη (σύνθετο κύριο κλειδί, ξένο κλειδί)
- **task:** Κωδικός εργασίας (σύνθετο κύριο κλειδί, ξένο κλειδί)

tbl_reminder: Το περιεχόμενο του πίνακα αυτού είναι οι υπενθυμίσεις που ορίζει η εφαρμογή ή ο χρήστης για τις επόμενες εργασίες που πρέπει να γίνουν σε όχημα του χρήστη. Αποτελείται από τις στήλες:

- **id:** Κωδικός της υπενθύμισης (κύριο κλειδί)
- **title:** Κείμενο που περιέχει η υπενθύμιση
- **date:** Ημερομηνία ορισμού υπενθύμισης
- **time:** Ώρα ορισμού υπενθύμισης

4.2.3 CarBook web services

Για να αντλήσει η εφαρμογή πληροφορίες από την βάση δεδομένων CarBook (βάση σε MySQL) είναι απαραίτητο να αναπτυχθεί διαδικτυακή υπηρεσία (web service). Η υπηρεσία αυτή συνδέεται στην βάση δεδομένων και εκτελεί κατάλληλα διαμορφωμένα ερωτήματα (αγγλικά: queries) ώστε να ληφθούν τα δεδομένα σε κατάλληλη μορφή, μεταφράσιμη από την εφαρμογή. Για την ανάπτυξη της διαδικτυακής υπηρεσίας, χρησιμοποιήθηκε το πλαίσιο εφαρμογής (αγγλικά: application framework) Spring Boot που υποστηρίζει η γλώσσα Java και κάνει την ανάπτυξη μιας υπηρεσίας ταχύτερη.

Στο πλαίσιο ανάπτυξης της παρούσας εφαρμογής έχει επιλεγεί η μορφή δεδομένων JSON. Όταν γίνει λήψη των πληροφορίων η τελική εφαρμογή χρησιμοποιώντας την κλάση Volley (βιβλιοθήκη που διευκολύνει την διασύνδεση των android εφαρμογών στο διαδίκτυο) «φιλτράρει» τα δεδομένα και τα χρησιμοποιεί καταλλήλως. Παρακάτω ακολουθεί ο πίνακας

με τα web services (**Get requests**) που αναπτύχθηκαν για τους σκοπούς της εφαρμογής CarBook (πελάτης).

Web Service	**Είσοδος**	**Έξοδος**
http://localhost:8080/api/carbrands	-	Επιστρέφει όλες τις μάρκες αυτοκινήτων που είναι καταχωρημένες στην βάση δεδομένων.
http:// localhost:8080/api/byname/selectedBrand	name (όνομα μάρκας)	Επιστρέφει τα δεδομένα για την μάρκα με όνομα «name»
http:// localhost:8080/api/carbrand/brands	brands (κωδικός μάρκας)	Επιστρέφει τα δεδομένα για την μάρκα με κωδικό «brands»
http://localhost:8080/api/bybrand?brand= getBrandCode	getBrandCode (κωδικός μάρκας)	Επιστρέφει τα δεδομένα για το μοντέλο αυτοκινήτου με κωδικό μάρκας «getBrandCode»
http://localhost:8080/api/bymodelyear/ modelName/year	modelName (όνομα μοντέλου) year (χρονιά κυκλοφορίας μοντέλου	Επιστρέφει τα δεδομένα για το μοντέλο αυτοκινήτου με «modelName» και χρονιά «year»
http://localhost:8080/api/cartask/taskCode	taskCode (κωδικός εργασίας)	Επιστρέφει τα δεδομένα για την εργασία με κωδικό «taskCode»
http://localhost :8080/api/cartasks/bytype?taskType=taskType	taskType (τύπος εργασίας)	Επιστρέφει όλες τις εργασίες που είναι καταχωρημένες στην βάση με τύπο εργασίας «taskType»
http://localhost:8080/api/workshops	-	Επιστρέφει όλα τα συνεργεία που είναι

		καταχωρημένα στην βάση
http://localhost:8080/api/workshop/code_w	code_w (κωδικός συνεργείου)	Επιστρέφει τα δεδομένα για το συνεργείο με κωδικό «code_w»
http://localhost:8080/api/wstype?codeW=codeW	codeW (κωδικός συνεργείου)	Επιστρέφει όλους τους τύπους εργασιών για το συνεργείο με κωδικό «codeW»
http://localhost:8080/api/authorizedworkshops?brands=id	id (κωδικός μάρκας)	Επιστρέφει όλα τα εξουσιοδοτημένα συνεργεία με βάση το «id» (κωδικό μάρκας)
http://localhost:8080/api/authorizedbrands?codeW=codeW	codeW (κωδικός συνεργείου	Επιστρέφει τις μάρκες για τις οποίες το συνεργείο είναι εξουσιοδοτημένο με βάση το «codeW»
http://localhost:8080/api/bymodel?modelCode=ModelCode	ModelCode (κωδικός μοντέλου)	Επιστρέφει όλες τις καταχωρήσεις στην βάση για το πρόγραμμα συντήρησης του μοντέλου με κωδικό «ModeCode»

Πίνακας 4.10 Πίνακας με τις μεθόδους της διαδικτυακής υπηρεσίας (web service)

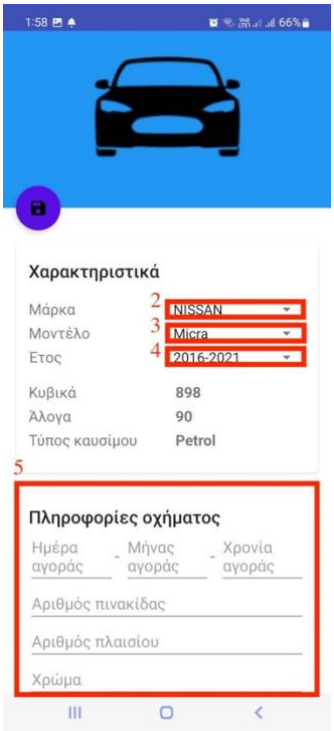
Κεφάλαιο 5: Οδηγίες χρήσης της εφαρμογής Carbook

Στο παρόν κεφάλαιο ακολουθούν συνοπτικά οι οδηγίες που πρέπει να ακολουθήσει ο χρήστης της εφαρμογής ώστε να αξιοποιήσει τις λειτουργίες που παρέχει.

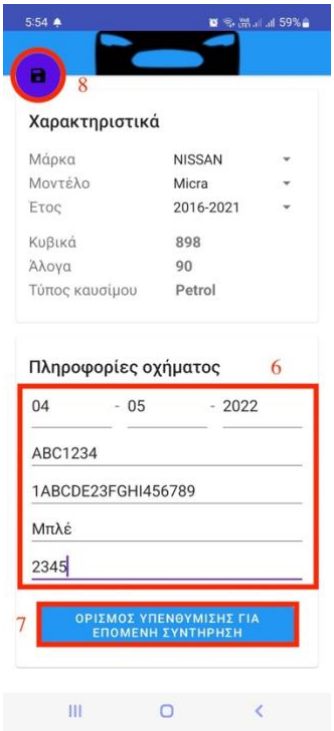
5.1 Προσθήκη οχήματος

- Επιλέγει ο χρήστης το εικονίδιο στο «συρτάρι εφαρμογών» της συσκευής για να εκκινήσει την εφαρμογή.
- Στην πρώτη οθόνη που εμφανίζεται, ο χρήστης επιλέγει το πλήκτρο που φαίνεται στην εικόνα 5.1 .
- Στην οθόνη που εμφανίζεται (εικόνα 5.2) επιλέγει με την σειρά την μάρκα, το μοντέλο και την χρονιά κυκλοφορίας του αυτοκινήτου. Όταν ο χρήστης επιλέγει μια μάρκα εμφανίζονται αυτόματα τα μοντέλα και οι αντίστοιχες χρονολογίες. Τέλος πληκτρολογεί τις πληροφορίες στα πεδία.
- Αν το επιθυμεί ο χρήστης, μπορεί να επιλέξει «Ορισμός υπενθύμισης για επόμενη συντήρηση» (εικόνα 5.3), ώστε η εφαρμογή να τον ειδοποιήσει όταν είναι κοντά το διάστημα συντήρησης (με βάση τα χιλιόμετρα που έχει καταχωρήσει στο πεδίο «οδόμετρο»).
- Το όχημα έχει καταχωρηθεί επιτυχώς (εικόνα 5.4).

Εικόνα 5.1:Αρχική σελίδα

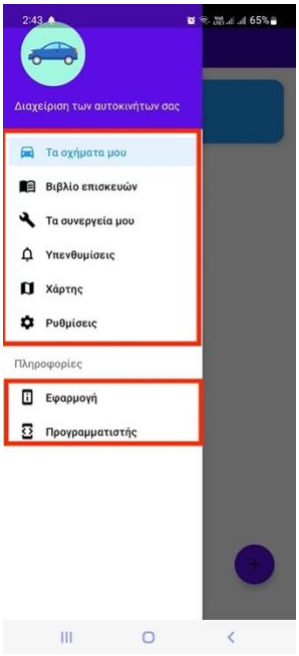


Εικόνα 5.2: Συμπλήρωση πεδίων



Εικόνα 5.3: Συμπλήρωση πεδίων, ορισμός υπενθύμισης και αποθήκευση

Εικόνα 5.4: Αποθηκευμένο όχημα



Εικόνα 5.5: Μενού λειτουργιών

5.2 Μενού επιλογής λειτουργιών

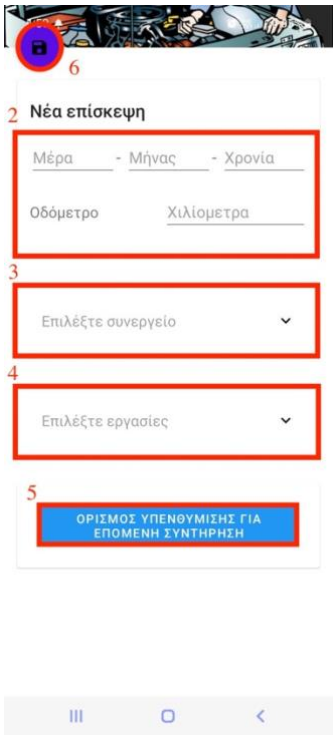
Το μενού λειτουργιών περιλαμβάνει όλες τις λειτουργίες σε ένα πλευρικό πάνελ. Ο χρήστης μπορεί να επιλέξει την οθόνη που θα εμφανιστεί. Όπως φαίνεται στην εικόνα 5.4 επιλέγοντας το πλήκτρο ☰ ανοίγει το πάνελ (εικόνα 5.5).

5.3 Βιβλίο επισκευών και διαχείριση επισκέψεων

Εικόνα 5.6: Βιβλίο επισκευών νέου αυτοκινήτου.

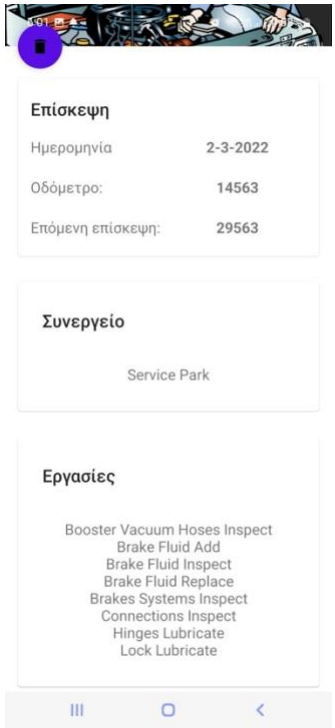
Στην παραπάνω οθόνη (εικόνα 5.6) εμφανίζεται το βιβλίο επισκευών για κάθε αυτοκίνητο του χρήστη. Ακολουθώντας τα βήματα με την σειρά όπως φαίνονται στις εικόνες 5.6, 5.7 και 5.8 , ο χρήστης μπορεί να αποθηκεύσει τις επισκέψεις στα συνεργεία με τις αντίστοιχες εργασίες που έγιναν, να έχει πρόσβαση σε αυτές όταν χρειάζεται και τέλος μπορεί να τις αφαιρέσει. Στην οθόνη της εικόνας 5.10 φαίνεται μια αποθηκευμένη επίσκεψη με τις αντίστοιχες εργασίες που έγιναν.

Εικόνα 5.7: Εισαγωγή νέας επίσκεψης.



Εικόνα 5.8: Εισαγωγή νέας επίσκεψης.

Εικόνα 5.9: Αποθηκευμένες επισκέψεις.

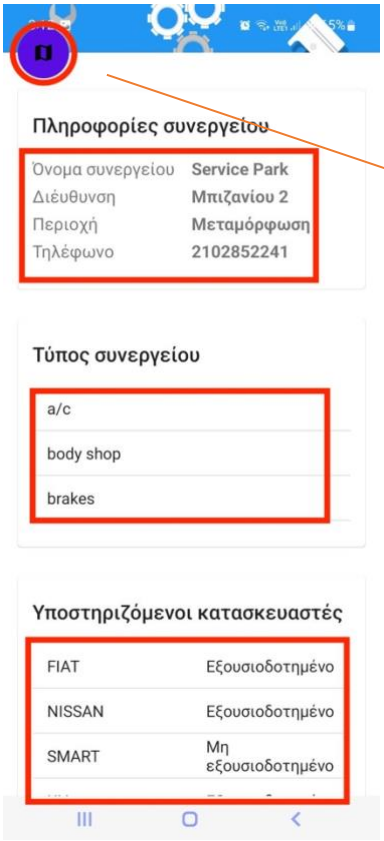


Εικόνα 5.10: Αποθηκευμένες εργασίες σε επίσκεψη.

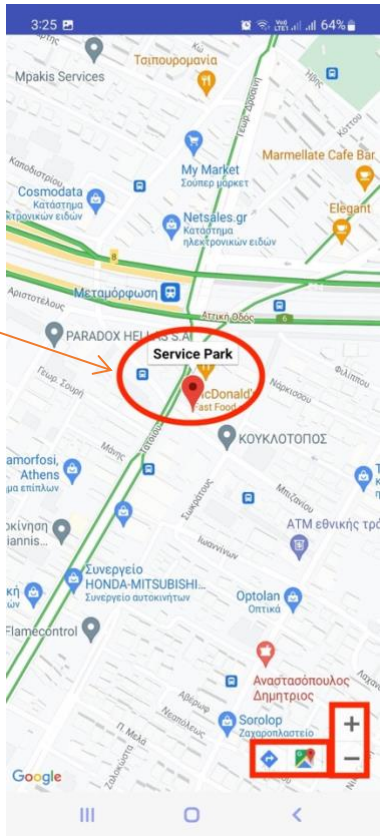
5.4 Τα συνεργεία μου

Στην οθόνη αυτή εμφανίζεται μια λίστα με συνεργεία που υποστηρίζουν τα οχήματα που έχει καταχωρήσει ο χρήστης στην εφαρμογή. Μόλις προστεθεί ένα όχημα εμφανίζονται τα αντίστοιχα συνεργεία.

Ο χρήστης επιλέγοντας ένα συνεργείο, μπορεί να μάθει περισσότερες πληροφορίες για το συνεργείο. Πιο συγκεκριμένα στην εικόνα 5.11 εμφανίζονται οι πληροφορίες για το συνεργείο (διεύθυνση, περιοχή, όνομα και τηλέφωνο). Παρακάτω φαίνεται ο τύπος εργασιών που πραγματοποιεί το συνεργείο καθώς και οι κατασκευαστές των οποίων τα αυτοκίνητα υποστηρίζονται. Επιπλέον ο χρήστης έχει την δυνατότητα να δει στον χάρτη που βρίσκεται το συνεργείο (εικόνα 5.12).



Εικόνα 5.11: Οθόνη πληροφορίων, τύποι εργασιών και υποστηριζόμενες μάρκες συνεργείου.



Εικόνα 5.12: Επιλεγμένο συνεργείο στον χάρτη.

Στην εικόνα 5.12 στην οθόνη του χάρτη εμφανίζεται η πινέζα της τοποθεσίας. Ο χρήστης μπορεί να εμφανίζει το όνομα του συνεργείου επιλέγοντας την πινέζα.

Πάνω στον χάρτη φαίνεται η κίνηση στις κεντρικές αρτηρίες και υπάρχουν επιπλέον επιλογές για τον χάρτη.

5.5 Υπενθυμίσεις

Η εφαρμογή υποστηρίζει την αυτόματο ορισμό υπενθυμίσεων για την ολοκλήρωση τον επερχόμενων εργασιών συντήρησης ή επισκευής. Όταν ο χρήστης καταχωρήσει ένα αυτοκίνητο, με βάση τα χιλιόμετρα στο οδόμετρο, δημιουργείται (αν το επιλέξει ο χρήστης) υπενθύμιση με βάση το αντίστοιχο χρονικό διάστημα. Παράλληλα ορίζονται υπενθυμίσεις με

κάθε νέα καταχώρηση επίσκεψης στο βιβλίο επισκευών με βάση τα χιλιόμετρα και τις προηγούμενες εργασίες που έχουν ολοκληρωθεί.

Στην εικόνα 5.13 βλέπουμε την αρχική οθόνη των υπενθυμίσεων. Ο χρήστης έχει την δυνατότητα να ορίσει χειροκίνητα υπενθυμίσεις επιλέγοντας το κατάλληλο πλήκτρο (εικόνα 5.14).

Στην εικόνα 5.15 φαίνεται η υπενθύμιση που έχει οριστεί αυτόματα από το σύστημα. Ο χρήστης έχει την δυνατότητα να βλέπει ποιες εργασίες πρέπει να γίνουν και σε ποια συνεργεία υποστηρίζονται.

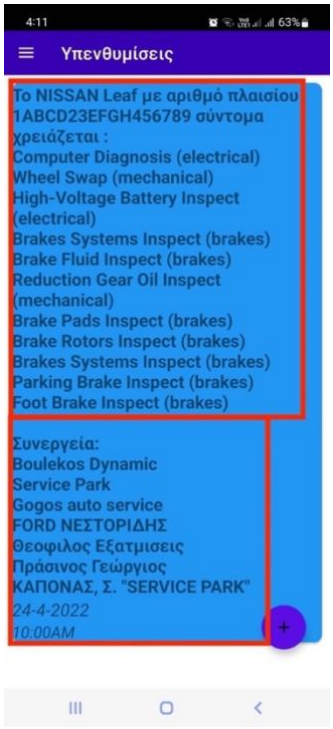
Στην εικόνα 5.16 ο χρήστης έχει την δυνατότητα να επεξεργαστεί την υπενθύμιση και να ορίσει άλλη ημερομηνία και ώρα για την ειδοποίηση.

Εικόνα 5.13: Αρχική οθόνη υπενθυμίσεων.

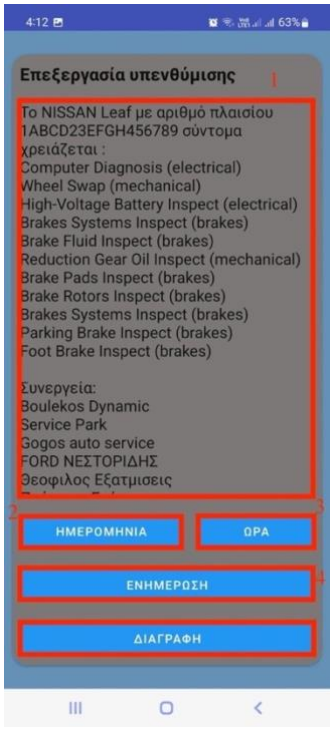
Εικόνα 5.14: Οθόνη ορισμού υπενθυμίσεων.

5.6 Χάρτης

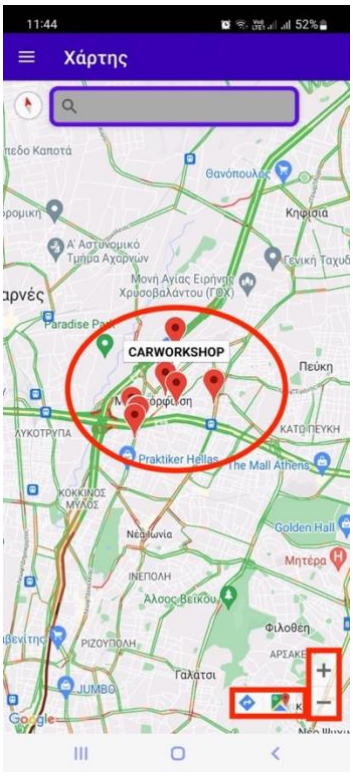
Η επιλογή «Χάρτης» δίνει την δυνατότητα στον χρήστη με μια ματιά να βλέπει τη λίστα με τα κατάλληλα συνεργεία για το όχημα του πάνω στον χάρτη (εικόνα 5.17). Η εφαρμογή υποστηρίζει αναζήτηση τοποθεσιών, ώστε ο χρήστης να έχει την δυνατότητα να δει τα κατάλληλα συνεργεία για το όχημα του με βάση την περιοχή ή την πόλη (εικόνα 5.18 και 5.19).



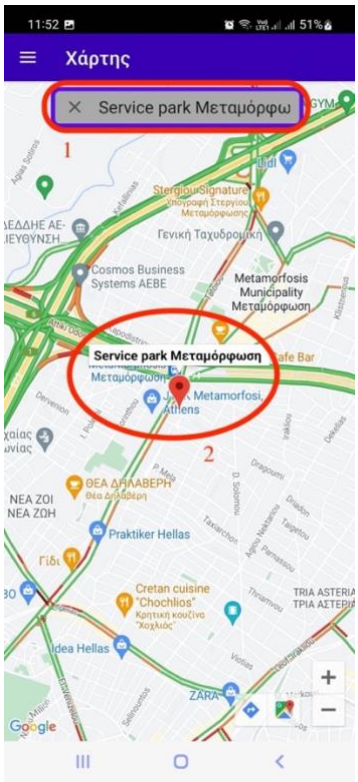
Εικόνα 5.15: Αρχική οθόνη υπενθυμίσεων με ορισμένες από το σύστημα υπενθυμίσεις.



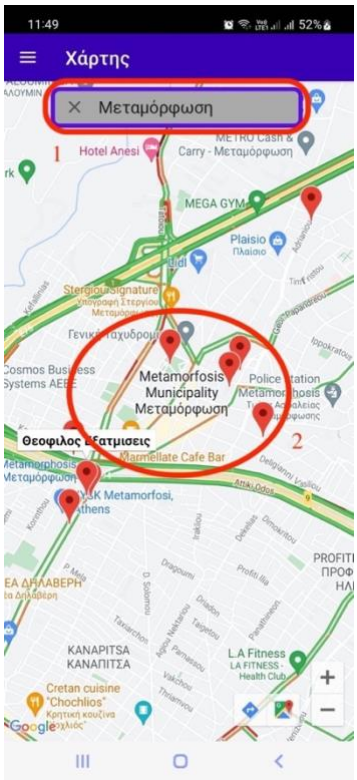
Εικόνα 5.16: Οθόνη επεξεργασίας υπενθυμίσεων.



Εικόνα 5.17: Αρχική οθόνη του χάρτη όπου φαίνονται τα κατάλληλα συνεργεία για τα οχήματα του χρήστη.



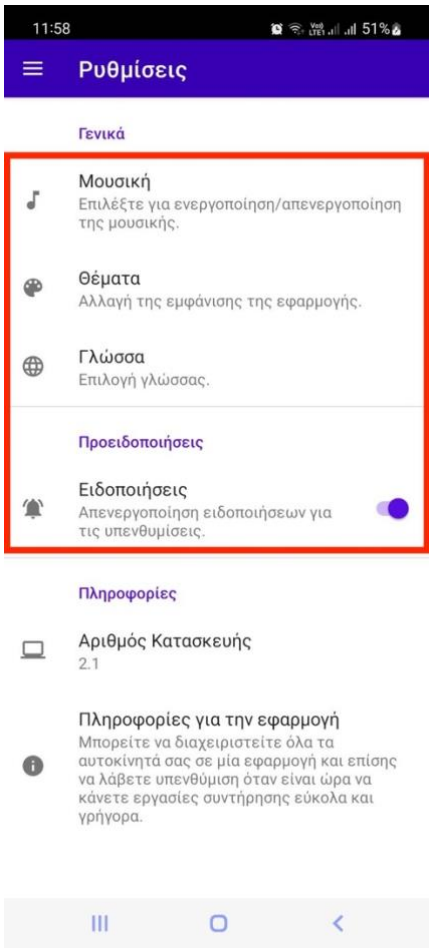
Εικόνα 5.18: Οθόνη χάρτη όταν έχει πραγματοποιηθεί αναζήτηση ενός συνεργείου.



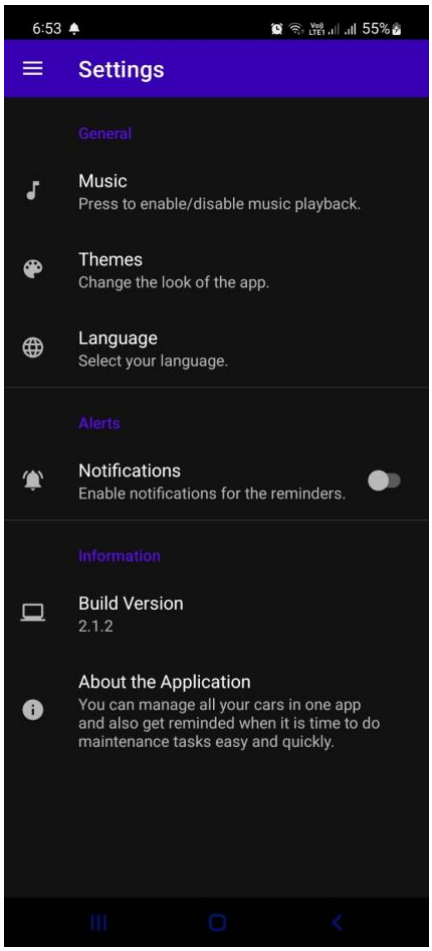
Εικόνα 5.19: Οθόνη χάρτη όταν έχει πραγματοποιηθεί αναζήτηση μιας περιοχής. Εμφανίζονται τα συνεργεία που είναι διαθέσιμα για τα οχήματα του χρήστη.

5.7 Ρυθμίσεις εφαρμογής

Η εφαρμογή CarBook υποστηρίζει βασικές ρυθμίσεις για την αλλαγή θέματος, αλλαγή γλώσσας σε ελληνικά ή αγγλικά και την ενεργοποίηση/απενεργοποίηση ειδοποιήσεων για τις υπενθυμίσεις (εικόνες 5.20 και 5.21).



Εικόνα 5.20: Οθόνη ρυθμίσεων στα ελληνικά και σε φωτεινό θέμα.



Εικόνα 5.21: Οθόνη ρυθμίσεων στα αγγλικά και σε σκοτεινό θέμα.

Παράρτημα Α: Κώδικας της εφαρμογής CarBook σε Java

MainActivity.java

```
package com.example.carbook.mainactivity;

 import android.content.SharedPreferences;
 import android.os.Bundle;
 import android.view.MenuItem;
 import android.widget.Toast;

 import androidx.annotation.NonNull;
 import androidx.appcompat.app.ActionBar;
 import androidx.appcompat.app.ActionBarDrawerToggle;
 import androidx.appcompat.app.AppCompatDelegate;
 import androidx.appcompat.widget.Toolbar;
 import androidx.core.view.GravityCompat;
 import androidx.drawerlayout.widget.DrawerLayout;
 import androidx.preference.PreferenceManager;

 import com.example.carbook.MapFragment;
 import com.example.carbook.R;
 import com.example.carbook.mycars.MyCarsFragment;
 import com.example.carbook.myworkshops.MyWorkshopsFragment;
 import com.example.carbook.reminders.RemindersFragment;
 import com.example.carbook.servicebook.ServiceBookFragment;
 import com.example.carbook.settings.MyAppCompat;
 import com.example.carbook.settings.SettingsFragment;
 import com.google.android.material.navigation.NavigationView;



 public class MainActivity extends MyAppCompat implements
NavigationView.OnNavigationItemSelectedListener {
    private DrawerLayout drawer;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        PreferenceManager.setDefaultValues(this, R.xml.fragment_settings,
false);
        Toolbar toolbar = findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);


        drawer = findViewById(R.id.drawer_layout);
        NavigationView navigationView = findViewById(R.id.nav_view);
        navigationView.setNavigationItemSelectedListener(this);

        ActionBarDrawerToggle toggle = new
ActionBarDrawerToggle(this,drawer,toolbar,

R.string.navigation_drawer_open,R.string.navigation_drawer_close );
        drawer.addDrawerListener(toggle);
        toggle.syncState();
```

```
        if(savedInstanceState == null) {  //When the app starts the mycar
page shows.

getSupportFragmentManager().beginTransaction().replace(R.id.fragment_conten
er, new MyCarsFragment()).commit();
            navigationView.setCheckedItem(R.id.nav_mycar);
        }

        loadSettings();
    }

    @Override
    public boolean onNavigationItemSelected(@NonNull MenuItem item) {
        //Open the fragments
        //addToBackStack function enables the user to press the back key
and return to the previous page.
        switch (item.getItemId()) {
            case R.id.nav_mycar:

getSupportFragmentManager().beginTransaction().replace(R.id.fragment_conten
er, new MyCarsFragment()).addToBackStack("fragmentStack").commit();
                break;
            case R.id.nav_servicebook:

getSupportFragmentManager().beginTransaction().replace(R.id.fragment_conten
er, new ServiceBookFragment()).addToBackStack("fragmentStack").commit();
                break;
            case R.id.nav_myworkshops:

getSupportFragmentManager().beginTransaction().replace(R.id.fragment_conten
er, new MyWorkshopsFragment()).addToBackStack("fragmentStack").commit();
                break;
            case R.id.nav_reminders:

getSupportFragmentManager().beginTransaction().replace(R.id.fragment_conten
er, new RemindersFragment()).addToBackStack("fragmentStack").commit();
                break;
            case R.id.nav_map:

getSupportFragmentManager().beginTransaction().replace(R.id.fragment_conten
er, new MapFragment()).addToBackStack("fragmentStack").commit();
                break;
            case R.id.nav_settings:

getSupportFragmentManager().beginTransaction().replace(R.id.fragment_conten
er, new SettingsFragment()).addToBackStack("fragmentStack").commit();
                //Setup preferences
                setupSharedPreferences();
                ActionBar titleS = getSupportActionBar();
                titleS.setTitle(getString(R.string.settings));  //Title of
the settings page
                setupPermissions();
                break;
            case R.id.nav_developer:
                openDialog();
                break;
            case R.id.nav_app:
                Toast.makeText(this, R.string.toast_info,
Toast.LENGTH_SHORT).show();
                break;
```

```
        }
        drawer.closeDrawer(GravityCompat.START);
        return true;
    }

    @Override
    public void onBackPressed() {  //If the menu drawer is open, it closes.
        if(drawer.isDrawerOpen(GravityCompat.START)) {
            drawer.closeDrawer(GravityCompat.START);
        } else {
            super.onBackPressed();
        }
    }

    public void openDialog(){
        DevDialog devDialog = new DevDialog();
        devDialog.show(getSupportFragmentManager(),"developer dialog");
    }

    private void setupSharedPreferences(){ //Get a reference to the default
shared preferences


    }

    private void setupPermissions(){

    }

    private void loadSettings(){
        SharedPreferences sp =
PreferenceManager.getDefaultSharedPreferences(this);
        String spTheme = sp.getString("theme", "false");
        if ("1".equals(spTheme)){

AppCompatDelegate.setDefaultNightMode(AppCompatDelegate.MODE_NIGHT_NO);
        }
        else if ("2".equals(spTheme)){

AppCompatDelegate.setDefaultNightMode(AppCompatDelegate.MODE_NIGHT_YES);
        }
        else if ("3".equals(spTheme)){

AppCompatDelegate.setDefaultNightMode(AppCompatDelegate.MODE_NIGHT_FOLLOW_S
YSTEM);
        }

    }

    @Override
    protected void onResume() {
        loadSettings();
        super.onResume();
    }

}
```

DevDialog.java

```
package com.example.carbook.mainactivity;

import android.app.Dialog;
import android.content.DialogInterface;
import android.os.Bundle;

import androidx.appcompat.app.AlertDialog;
import androidx.appcompat.app.AppCompatDialogFragment;

import com.example.carbook.R;

public class DevDialog extends AppCompatDialogFragment {

    @Override
    public Dialog onCreateDialog( Bundle savedInstanceState) {
        AlertDialog.Builder builder = new
AlertDialog.Builder(getActivity());
        builder.setTitle(R.string.dialog_title)
                .setMessage(getString(R.string.info1)

+System.lineSeparator()+getString(R.string.info2)+System.lineSeparator()+ge
tString(R.string.info3))
                .setPositiveButton(R.string.exit, new
DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialogInterface,
int i) {

                    }
                });
        return builder.create();

    }
}
```

RequestSingleton.java

```
package com.example.carbook.mainactivity;

import android.content.Context;

import com.android.volley.Request;
import com.android.volley.RequestQueue;
import com.android.volley.toolbox.Volley;


public class RequestSingleton {

    private static RequestSingleton instance;
    private RequestQueue requestQueue;
    private static Context ctx;

    private RequestSingleton(Context context){
        ctx = context;
        requestQueue = getRequestQueue();
    }

    public static synchronized RequestSingleton getInstance(Context
```

```
context) {
        if (instance == null) {
            instance = new RequestSingleton(context);
        }
        return instance;
    }

    public RequestQueue getRequestQueue() {
        if (requestQueue == null) {
            //getApplicationContext is key, it keeps you from leaking the
            //Activity or BroadcastReceiver if it is passed
            requestQueue =
Volley.newRequestQueue(ctx.getApplicationContext());
        }
        return requestQueue;
    }

    public <T> void addToRequestQueue(Request<T> req){
         getRequestQueue().add(req);
    }
}
```

dbCarManager.java

```
package com.example.carbook.mycars;

import android.content.ContentValues;
import android.content.Context;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;

import androidx.annotation.Nullable;

public class dbCarManager extends SQLiteOpenHelper {
    private static String dbname = "myCars";

    public dbCarManager(@Nullable Context context) {
        super(context, dbname, null, 1);
    }

    @Override
    public void onCreate(SQLiteDatabase sqLiteDatabase) {
//sql query to insert data in sqllite
        String query = "create table tbl_cars(id integer primary key
autoincrement,Brand text,Model text,Year text,Cubic text,Horsepower
text,Fuel text ,DateDay text, DateMonth text, DateYear text,Chassis
text,Plate text, Color text, ModelCode integer)";
        sqLiteDatabase.execSQL(query);
        String query1 ="create table tbl_car_visit(visit_code integer
primary key autoincrement, mycar_code integer, speedometer
integer,next_service integer,w_code integer,p_date text,foreign key
(mycar_code) references tbl_cars (id))";
        sqLiteDatabase.execSQL(query1);
        String query2 ="create table tbl_car_service(visit_code integer,
task text, primary key(visit_code,task), foreign key (visit_code)
references tbl_car_visit (visit_code))";
        sqLiteDatabase.execSQL(query2);
    }
```

```
    @Override
    public void onUpgrade(SQLiteDatabase sqLiteDatabase, int i, int i1) {

        String query = "DROP TABLE IF EXISTS tbl_cars";
//sql query to check table with the same name or not
        sqLiteDatabase.execSQL(query);
//executes the sql command
        String query1 = "DROP TABLE IF EXISTS tbl_car_visit";
//sql query to check table with the same name or not
        sqLiteDatabase.execSQL(query1);
        onCreate(sqLiteDatabase);

    }

     //CRUD methods for tbl_cars
    public String addCar(String Brand, String Model, String Year, String
Cubic,String Horsepower, String Fuel,  String DateDay, String DateMonth,
String DateYear, String Chassis, String Plate, String Color, Long
ModelCode){
        SQLiteDatabase database = this.getReadableDatabase();

        ContentValues contentValues = new ContentValues();
        contentValues.put("Brand", Brand);
//Inserts  data into sqllite database
        contentValues.put("Model", Model);
        contentValues.put("Year", Year);
        contentValues.put("Cubic", Cubic);
        contentValues.put("Horsepower", Horsepower);
        contentValues.put("Fuel", Fuel);
        contentValues.put("DateDay", DateDay);
        contentValues.put("DateMonth", DateMonth);
        contentValues.put("DateYear", DateYear);
        contentValues.put("Chassis", Chassis);
//Inserts  data into sqllite database
        contentValues.put("Plate", Plate);
        contentValues.put("Color", Color);
        contentValues.put("ModelCode", ModelCode);

        float result = database.insert("tbl_cars", null, contentValues);
//returns -1 if data successfully inserts into database

        if (result == -1) {
            return "Failed";
        } else {
            return "Successfully inserted";
        }
    }

    public String updatecars (String Model,String Brand, String newModel,
String Year, String Cubic,String Horsepower, String Fuel, String DateDay,
String DateMonth, String DateYear, String Chassis, String Plate, String
Color, Long ModelCode){
        SQLiteDatabase database = this.getWritableDatabase();

        ContentValues contentValues = new ContentValues();
        contentValues.put("Brand", Brand);
//Inserts  data into sqllite database
        contentValues.put("Model", newModel);
        contentValues.put("Year", Year);
        contentValues.put("Cubic", Cubic);
        contentValues.put("Horsepower", Horsepower);
```

```
        contentValues.put("Fuel", Fuel);
        contentValues.put("DateDay", DateDay);
        contentValues.put("DateMonth", DateMonth);
        contentValues.put("DateYear", DateYear);
        contentValues.put("Chassis", Chassis);
//Inserts  data into sqllite database
        contentValues.put("Plate", Plate);
        contentValues.put("Color", Color);
        contentValues.put("ModelCode", ModelCode);

        float result = database.update("tbl_cars", contentValues,
"Model=?",new String[]{Model});    //returns -1 if data successfully
inserts into database

        if (result == -1) {
            return "Failed";
        } else {
            return "Successfully inserted";
        }
    }

    public String deletecar(Long id){
        SQLiteDatabase database = this.getWritableDatabase();

        float result = database.delete("tbl_cars", "id=?", new
String[]{id.toString()});  //returns -1 if data successfully inserts into
database
        if (result == -1) {
            return "Failed";
        } else {
            return "Successfully deleted";
        }


    }

    public Cursor readallcars() {
        SQLiteDatabase database = this.getWritableDatabase();
        String query = "select * from tbl_cars order by id desc";
//Sql query to  retrieve  data from the database
        Cursor cursor = database.rawQuery(query, null);
        return cursor;
    }

    public Cursor getMyCarId(String d){

        SQLiteDatabase database = this.getWritableDatabase();
        String query = "select id from tbl_cars where DateYear = " + d;
//Sql query to  retrieve  data from the database
        Cursor cursor = database.rawQuery(query,null);
            return cursor;


    }



    public Cursor getMyCarModel(Long id){

        SQLiteDatabase database = this.getWritableDatabase();
```

```
        String query = "select Model,Brand,ModelCode from tbl_cars where id
= "+id;                                 //Sql query to  retrieve  data from
the database
        Cursor cursor = database.rawQuery(query, null);
        return cursor;

    }

    public Cursor getMyCarModel1(){

        SQLiteDatabase database = this.getWritableDatabase();
        String query = "select Brand from tbl_cars";
//Sql query to  retrieve  data from the database
        Cursor cursor = database.rawQuery(query, null);
        return cursor;

    }

    public Cursor carIsEmpty() {
        SQLiteDatabase database = this.getWritableDatabase();
        String query = "select count(*) from tbl_cars";
        Cursor cursor = database.rawQuery(query, null);
        return cursor;
    }



    //CRUD methods for tbl_car_carvisit
    public String addVisit(Long mycar_code, Long speedometer, Long
next_service,Long w_code, String p_date){
        SQLiteDatabase database = this.getReadableDatabase();

        ContentValues contentValues = new ContentValues();  //Inserts  data
into sqllite database
        contentValues.put("mycar_code", mycar_code);
        contentValues.put("speedometer", speedometer);
        contentValues.put("next_service", next_service);
        contentValues.put("w_code", w_code);
        contentValues.put("p_date", p_date);

        float result = database.insert("tbl_car_visit", null,
contentValues);    //returns -1 if data fail to insert into database

        if (result == -1) {
            return "Failed";
        } else {
            return "Successfully inserted";
        }
    }

    public String updateVisit (Long visit_code, Long mycar_code, Long
speedometer, Long next_service,Long w_code, String p_date){
        SQLiteDatabase database = this.getWritableDatabase();

        ContentValues contentValues = new ContentValues();
        contentValues.put("visit_code", visit_code);
//Inserts  data into sqllite database
        contentValues.put("mycar_code", mycar_code);
        contentValues.put("speedometer", speedometer);
        contentValues.put("next_service", next_service);
        contentValues.put("w_code", w_code);
```

```
        contentValues.put("p_date", p_date);

        float result = database.update("tbl_car_visit", contentValues,
"visit_code=?",new String[]{visit_code.toString()});    //returns -1 if
data successfully inserts into database

        if (result == -1) {
            return "Failed";
        } else {
            return "Successfully inserted";
        }
    }

    public String deleteVisit(Long visit_code){
        SQLiteDatabase database = this.getWritableDatabase();
        float result = database.delete("tbl_car_visit", "visit_code=?", new
String[]{visit_code.toString()});  //returns -1 if data successfully
inserts into database
        if (result == -1) {
            return "Failed";
        } else {
            return "Successfully deleted";
        }




    }

    public Cursor readallVisits() {
        SQLiteDatabase database = this.getWritableDatabase();
        String query = "select * from tbl_car_visit order by visit_code
desc";                              //Sql query to  retrieve  data from
the database
        Cursor cursor = database.rawQuery(query, null);
        return cursor;
    }

    public Cursor readVisit(Long id) {
        SQLiteDatabase database = this.getWritableDatabase();
        String query = "select * from tbl_car_visit where visit_code ="
+id+"  order by visit_code desc";                                  //Sql query
to  retrieve  data from the database
        Cursor cursor = database.rawQuery(query, null);
        return cursor;
    }


    public Cursor readModelVisits(Long id) {
        SQLiteDatabase database = this.getWritableDatabase();
        String query = "select * from tbl_car_visit where mycar_code ="
+id+"  order by visit_code desc";                                  //Sql query
to  retrieve  data from the database
        Cursor cursor = database.rawQuery(query, null);
        return cursor;
    }

    //Get the visit code
    public Cursor getVisitCode(Long odometer) {
        SQLiteDatabase database = this.getWritableDatabase();
        String query = "select visit_code from tbl_car_visit where
```

```
speedometer =" +odometer;                                    //Sql query to
retrieve  data from the database
        Cursor cursor = database.rawQuery(query, null);
        return cursor;
    }

    public Cursor visitIsEmpty() {
        SQLiteDatabase database = this.getWritableDatabase();
        String query = "select count(*) from tbl_car_visit";
        Cursor cursor = database.rawQuery(query, null);
        return cursor;
    }

    //CRUD methods for tbl_car_carvisit
    public String addService (Long visit_code, String task){
        SQLiteDatabase database = this.getReadableDatabase();

        ContentValues contentValues = new ContentValues();
        contentValues.put("visit_code", visit_code);
//Inserts  data into sqllite database
        contentValues.put("task", task);

        float result = database.insert("tbl_car_service", null,
contentValues);    //returns -1 if data successfully inserts into database

        if (result == -1) {
            return "Failed";
        } else {
            return "Successfully inserted";
        }
    }

    public String updateService (Long visit_code, Long task_code){
        SQLiteDatabase database = this.getWritableDatabase();

        ContentValues contentValues = new ContentValues();
        contentValues.put("visit_code", visit_code);
//Inserts  data into sqllite database
        contentValues.put("task_code", task_code);

        float result = database.update("tbl_car_service", contentValues,
"visit_code=?",new String[]{visit_code.toString()});    //returns -1 if
data successfully inserts into database

        if (result == -1) {
            return "Failed";
        } else {
            return "Successfully inserted";
        }
    }

    public String deleteService(Long visit_code){
        SQLiteDatabase database = this.getWritableDatabase();
        float result = database.delete("tbl_car_service", "visit_code=?",
new String[]{visit_code.toString()});  //returns -1 if data successfully
inserts into database
        if (result == -1) {
            return "Failed";
        } else {
            return "Successfully deleted";
        }
```

```
    }

    public Cursor readOneCarServices(long id) {
        SQLiteDatabase database = this.getWritableDatabase();
        String query = "select * from tbl_car_service where visit_code="+
id +" order by visit_code desc";                                    //Sql query
to  retrieve  data from the database
        Cursor cursor = database.rawQuery(query, null);
        return cursor;
    }




}
```

myCarAdapter.java

```
package com.example.carbook.mycars;

import android.annotation.SuppressLint;
import android.app.Activity;
import android.content.Context;
import android.content.Intent;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;

import androidx.annotation.NonNull;
import androidx.recyclerview.widget.RecyclerView;

import com.example.carbook.R;

import java.util.ArrayList;

public class myCarAdapter extends
RecyclerView.Adapter<myCarAdapter.myviewholder> {

    ArrayList<MyCarModel> dataholder = new ArrayList<MyCarModel>();
    Context context;
    Activity activity;
    //array list to hold the cars

    public myCarAdapter(Activity activity, Context context,
ArrayList<MyCarModel> dataholder) {
        this.dataholder = dataholder;
        this.context=context;
        this.activity=activity;
    }


    @NonNull
    @Override
    public myviewholder onCreateViewHolder(@NonNull ViewGroup parent, int
viewType) {
        View view =
LayoutInflater.from(parent.getContext()).inflate(R.layout.my_car_row,
parent, false);
        LayoutInflater layoutInflater = LayoutInflater.from(context);
```

```
        //inflates the xml file in recyclerview
        return new myviewholder(view);
    }

    @Override
    public void onBindViewHolder(@NonNull myCarAdapter.myviewholder holder,
@SuppressLint("RecyclerView") int position) {

        //Binds the single car objects to recycler view
        holder.mModel.setText(dataholder.get(position).getMyC_model());
        holder.mBrand.setText(dataholder.get(position).getMyC_brand());

        holder.itemView.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Intent intent = new Intent(context,
MyCarsViewActivity.class);
                intent.putExtra("model",
String.valueOf(dataholder.get(position).getMyC_model()));
                intent.putExtra("brand",
String.valueOf(dataholder.get(position).getMyC_brand()));
                intent.putExtra("year",
String.valueOf(dataholder.get(position).getMyC_year()));
                intent.putExtra("cubic",
String.valueOf(dataholder.get(position).getMyC_cubic()));
                intent.putExtra("horsepower",
String.valueOf(dataholder.get(position).getMyC_horsepower()));
                intent.putExtra("fueltype",
String.valueOf(dataholder.get(position).getMyC_fuelType()));
                intent.putExtra("dateday",
String.valueOf(dataholder.get(position).getPurchase_date_day()));
                intent.putExtra("datemonth",
String.valueOf(dataholder.get(position).getGetPurchase_date_month()));
                intent.putExtra("dateyear",
String.valueOf(dataholder.get(position).getPurchase_date_year()));
                intent.putExtra("chassis",
String.valueOf(dataholder.get(position).getChassis_number()));
                intent.putExtra("plate",
String.valueOf(dataholder.get(position).getPlate_number()));
                intent.putExtra("color",
String.valueOf(dataholder.get(position).getColor()));
                intent.putExtra("modelcode",
dataholder.get(position).getModel_code());
                //Starts the new activity to edit car
                activity.startActivity(intent);
            }
        });

    }

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

    class myviewholder extends RecyclerView.ViewHolder {

        TextView mModel, mBrand;
```

```
        public myviewholder(@NonNull View itemView) {
            super(itemView);

            mModel = itemView.findViewById(R.id.tv_model);
//holds the reference of the materials to show data in recyclerview
            mBrand = itemView.findViewById(R.id.tv_brand);

        }
    }
}
```

MyCarModel.java

```
package com.example.carbook.mycars;

public class MyCarModel {
    //MyCarModel class is used to set and get the data from the database

    public Long getId() {
        return id;
    }

    public void setId(Long id) {
        this.id = id;
    }

    Long id;
    String myC_brand;
    String myC_model;
    String myC_year;
    String myC_cubic;
    String myC_horsepower;
    String myC_fuelType;
    String purchase_date_day;
    String purchase_date_month;
    String purchase_date_year;
    String chassis_number;
    String plate_number;
    String color;
    Long model_code;



    public MyCarModel(Long id, String myC_code, String myC_model, String
myC_year, String myC_cubic, String myC_horsepower, String myC_fuelType,
String purchase_date_day, String purchase_date_month, String
purchase_date_year, String chassis_number, String plate_number, String
color, Long model_code){
        this.id = id;
        this.myC_brand = myC_code;
        this.myC_model = myC_model;
        this.myC_year = myC_year;
        this.myC_cubic = myC_cubic;
        this.myC_horsepower = myC_horsepower;
        this.myC_fuelType = myC_fuelType;
        this.purchase_date_day = purchase_date_day;
        this.purchase_date_month = purchase_date_month;
        this.purchase_date_year = purchase_date_year;
        this.chassis_number = chassis_number;
```

```
        this.plate_number = plate_number;
        this.color = color;
        this.model_code = model_code;

    }

    public Long getModel_code() {
        return model_code;
    }

    public void setModel_code(Long model_code) {
        this.model_code = model_code;
    }

    public String getPurchase_date_day() {
        return purchase_date_day;
    }

    public void setPurchase_date_day(String purchase_date_day) {
        this.purchase_date_day = purchase_date_day;
    }

    public String getGetPurchase_date_month() {
        return purchase_date_month;
    }

    public void setGetPurchase_date_month(String getPurchase_date_month) {
        this.purchase_date_month = getPurchase_date_month;
    }

    public String getMyC_year() {
        return myC_year;
    }

    public String getMyC_cubic() {
        return myC_cubic;
    }

    public String getMyC_horsepower() {
        return myC_horsepower;
    }

    public String getMyC_fuelType() {
        return myC_fuelType;
    }

    public String getMyC_brand() { return myC_brand; }

    public String getMyC_model() { return myC_model; }

    public String getPurchase_date_year() { return purchase_date_year; }

    public String getChassis_number() { return chassis_number; }

    public String getPlate_number() { return plate_number; }

    public String getColor() { return color; }

    public void setColor(String color) { this.color = color; }
}
```

MyCarsActivity.java

```
package com.example.carbook.mycars;
import android.app.AlarmManager;
import android.app.PendingIntent;
import android.content.Context;
import android.content.Intent;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Spinner;
import android.widget.TextView;
import android.widget.Toast;

import androidx.appcompat.app.ActionBar;
import androidx.appcompat.app.AppCompatActivity;

import com.android.volley.Request;
import com.android.volley.Response;
import com.android.volley.VolleyError;
import com.android.volley.toolbox.JsonArrayRequest;
import com.android.volley.toolbox.JsonObjectRequest;
import com.example.carbook.R;
import com.example.carbook.mainactivity.MainActivity;
import com.example.carbook.mainactivity.RequestSingleton;
import com.example.carbook.reminders.AlarmBroadcast;
import com.example.carbook.reminders.dbManager;
import com.example.carbook.servicebook.VisitScheduleModel;
import
com.google.android.material.floatingactionbutton.FloatingActionButton;

import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

import java.text.DateFormat;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;

public class MyCarsActivity extends AppCompatActivity implements
AdapterView.OnItemSelectedListener {


    //Spinners declaration
    Spinner myCarBrand, myCarModel, myCarYear;
    TextView myCarCubic,
myCarHorsePower,myCarFuelType,myCarModelCodeInvisible,tvHiddenTasks,tvHidde
nWS;
    EditText myCarDateDay,myCarDateMonth,myCarDateYear, myCarPlate,
myCarChassis, myCarColor, myCarOdometer;
    FloatingActionButton myCarSave;
    Button setReminder;
```

```
    String year,pDate,pDate1;
    Long ModelCode;
    Boolean notification, bool;
    int min_month,p_day, p_month,p_year;
    long nextService;

    //Arraylists and Array adapters for spinners
    ArrayList<String> brandsList = new ArrayList<>();
    ArrayList<String> modelsList = new ArrayList<>();
    ArrayList<String> yearList = new ArrayList<>();
    ArrayList<VisitScheduleModel> visitScheduleList = new ArrayList<>();

    ArrayAdapter<String> brandsAdapter;
    ArrayAdapter<String> modelsAdapter;
    ArrayAdapter<String> yearAdapter;


    String modelName, selectedBrand;


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_my_cars);
        ActionBar titleS = getSupportActionBar();
        titleS.hide();

        //access the data form the input fields and spinners
        myCarBrand = findViewById(R.id.mycar_brand);
        myCarModel = findViewById(R.id.mycar_model);
        myCarYear = findViewById(R.id.mycar_year);

        myCarCubic = findViewById(R.id.mycar_cubic);
        myCarHorsePower = findViewById(R.id.mycar_horsepower);
        myCarFuelType = findViewById(R.id.mycar_fueltype);
        myCarModelCodeInvisible = findViewById(R.id.invisible_model_code);
        tvHiddenTasks = findViewById(R.id.invisible_tasks);
        tvHiddenWS = findViewById(R.id.invisible_ws);


        myCarDateDay = findViewById(R.id.mycar_date_day);
        myCarDateMonth = findViewById(R.id.mycar_date_month);
        myCarDateYear = findViewById(R.id.mycar_date_year);
        myCarPlate =  findViewById(R.id.mycar_plate);
        myCarChassis = findViewById(R.id.mycar_chassis);
        myCarColor = findViewById(R.id.mycar_color);
        myCarOdometer = findViewById(R.id.mycar_odometer);

        myCarSave =  findViewById(R.id.mycar_save);
        setReminder = findViewById(R.id.set_rem_btn);

        //Configuration of the brands spinner from db data using Volley

        String url = "http://192.168.1.3:8080/api/carbrands"; //For android
phone

        JsonArrayRequest jsonArrayRequest = new
JsonArrayRequest(Request.Method.GET, url, null, new
Response.Listener<JSONArray>() {
            @Override
            public void onResponse(JSONArray response) {
```

```
                try {
                    JSONArray jsonArray = new
JSONArray(response.toString());
                    for(int i=0; i<jsonArray.length(); i++){
                        JSONObject jsonObject = jsonArray.getJSONObject(i);
                        String Brand = jsonObject.optString("name");

                        brandsList.add(Brand);
                        brandsAdapter = new
ArrayAdapter<>(MyCarsActivity.this,
                                android.R.layout.simple_spinner_item,
brandsList);

brandsAdapter.setDropDownViewResource(android.R.layout.simple_spinner_dropd
own_item);
                        myCarBrand.setAdapter(brandsAdapter);
                    }
                } catch (JSONException e) {
                    e.printStackTrace();
                }

            }
        }, new Response.ErrorListener() {
            @Override
            public void onErrorResponse(VolleyError error) {

                String errormsg;
                errormsg=
VolleyErrorHelper.getMessage(error,MyCarsActivity.this);
                Toast.makeText(getApplicationContext(), errormsg,
Toast.LENGTH_SHORT).show();

            }
        });

        //Add the json request object to the request queue

RequestSingleton.getInstance(this).addToRequestQueue(jsonArrayRequest);
        myCarBrand.setOnItemSelectedListener(this);


        setReminder.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {

                String Brand =
myCarBrand.getSelectedItem().toString().trim(); //For the id repeat the
above requests
                String DateDay = myCarDateDay.getText().toString().trim();
                String DateMonth =
myCarDateMonth.getText().toString().trim();
                String DateYear =
myCarDateYear.getText().toString().trim();
                String Plate = myCarPlate.getText().toString().trim();
                String Chassis = myCarChassis.getText().toString().trim();
                String Color = myCarColor.getText().toString().trim();
                String od = myCarOdometer.getText().toString().trim();
```

```
                if (Brand.isEmpty() || DateDay.isEmpty() ||
DateMonth.isEmpty() || DateYear.isEmpty() || Plate.isEmpty() ||
Chassis.isEmpty() || Color.isEmpty() ||
(myCarOdometer.getText().toString().trim()).equals("") ) {
                    Toast.makeText(getApplicationContext(),
getString(R.string.car_all_fields), Toast.LENGTH_SHORT).show();
                    //shows the toast if input field is empty
                } else if ((!DateYear.matches("\\d+(?:\\.\\d+)?")) ||
(!DateDay.matches("\\d+(?:\\.\\d+)?")) ||
(!DateMonth.matches("\\d+(?:\\.\\d+)?")) ||
(!od.matches("\\d+(?:\\.\\d+)?"))) {

                    Toast.makeText(getApplicationContext(),
getString(R.string.Wrong_data_input_msg_mycars),
Toast.LENGTH_SHORT).show();

                } else {

                    Long Odometer =
Long.parseLong(myCarOdometer.getText().toString().trim());
                    p_day =
Integer.parseInt(myCarDateDay.getText().toString().trim());
                    p_month =
Integer.parseInt(myCarDateMonth.getText().toString().trim());
                    p_year =
Integer.parseInt(myCarDateYear.getText().toString().trim());

                    if(Odometer==0L || Odometer<5000) {
                        String url =
"http://192.168.1.3:8080/api/bymodel?modelCode=" + ModelCode; //For android
phone

                    JsonArrayRequest jsonArrayRequest = new
JsonArrayRequest(Request.Method.GET, url, null, new
Response.Listener<JSONArray>() {
                        @Override
                        public void onResponse(JSONArray response) {

                            try {
                                JSONArray jsonArray = new
JSONArray(response.toString());
                                Long min = 3000000L;
                                min_month = 1000;
                                for (int i = 0; i < jsonArray.length();
i++) {
                                    JSONObject jsonObject =
jsonArray.getJSONObject(i);
                                    Long taskCode =
jsonObject.optLong("taskCode");
                                    Long kilometers =
jsonObject.optLong("kilometers");
                                    int months =
jsonObject.optInt("months");
                                    if (kilometers < min) {
                                        min = kilometers;
                                        min_month = months;
                                    }
                                    JSONObject jsonObject1 =
jsonObject.getJSONObject("vs");
                                    Long sCode =
jsonObject1.getLong("sCode");
```

```
                                    Long modelCode2 =
jsonObject1.getLong("modelCode");
                                    visitScheduleList.add(new
VisitScheduleModel(sCode, modelCode2, taskCode, kilometers, months));
                                }


                                for (int i = 0; i <
visitScheduleList.size(); i++) {

                                    int nextTasks = (int) (min /
visitScheduleList.get(i).getKilometers());
                                    nextTasks = Math.round(nextTasks);
//Rounded towards up if nextTasks==2,9 -> 2 rounded


                                    if (nextTasks == 1) {

                                        String url =
"http://192.168.1.3:8080/api/cartask/" +
visitScheduleList.get(i).getTaskCode();
                                        int finalI = i;
                                        Long finalMin = min;
                                        JsonObjectRequest jsonObjectRequest
= new JsonObjectRequest(Request.Method.GET, url, null, new
Response.Listener<JSONObject>() {
                                            @Override
                                            public void
onResponse(JSONObject response) {

                                                if
(finalMin.equals(visitScheduleList.get(finalI).getKilometers())) {  //First
service
                                                    String taskPart =
response.optString("taskPart");
                                                    String taskType =
response.optString("taskType");
                                                    String tvText =
tvHiddenTasks.getText().toString().trim();
                                                    String stringBuilder =
tvText + " \n" + taskPart + " (" + taskType + ")";

tvHiddenTasks.setText(stringBuilder);


                                                }

                                            }
                                        }, new Response.ErrorListener() {
                                            @Override
                                            public void
onErrorResponse(VolleyError error) {
                                                String errormsg;
                                                errormsg =
VolleyErrorHelper.getMessage(error, MyCarsActivity.this);

Toast.makeText(MyCarsActivity.this, errormsg, Toast.LENGTH_SHORT).show();
                                            }
                                        });

RequestSingleton.getInstance(MyCarsActivity.this).addToRequestQueue(jsonObj
```

```
ectRequest);
                                    }

                                }
                                String tvText =
tvHiddenTasks.getText().toString().trim();
                                String stringBuilder = tvText + "\n:";
                                tvHiddenTasks.setText(stringBuilder);

                            } catch (JSONException e) {
                                e.printStackTrace();
                            }

                        }
                    }, new Response.ErrorListener() {
                        @Override
                        public void onErrorResponse(VolleyError error) {

                            String errormsg;
                            errormsg = VolleyErrorHelper.getMessage(error,
MyCarsActivity.this);
                            Toast.makeText(MyCarsActivity.this, errormsg,
Toast.LENGTH_SHORT).show();

                        }
                    });


                    //Add the json request object to the request queue

RequestSingleton.getInstance(MyCarsActivity.this).addToRequestQueue(jsonArr
ayRequest);
                    }


                    //Request from database
                    String url =
"http://192.168.1.3:8080/api/bymodel?modelCode=" + ModelCode; //For android
phone

                    JsonArrayRequest jsonArrayRequest = new
JsonArrayRequest(Request.Method.GET, url, null, new
Response.Listener<JSONArray>() {
                        @Override
                        public void onResponse(JSONArray response) {

                            try {
                                JSONArray jsonArray = new
JSONArray(response.toString());
                                Long min = 3000000L;
                                min_month = 1000;
                                for (int i = 0; i < jsonArray.length();
i++) {
                                    JSONObject jsonObject =
jsonArray.getJSONObject(i);
                                    Long taskCode =
jsonObject.optLong("taskCode");
                                    Long kilometers =
jsonObject.optLong("kilometers");
                                    int months =
jsonObject.optInt("months");
```

```
                                    if (kilometers < min) {
                                        min = kilometers;
                                        min_month = months;
                                    }
                                    JSONObject jsonObject1 =
jsonObject.getJSONObject("vs");
                                    Long sCode =
jsonObject1.getLong("sCode");
                                    Long modelCode2 =
jsonObject1.getLong("modelCode");
                                    visitScheduleList.add(new
VisitScheduleModel(sCode, modelCode2, taskCode, kilometers, months));
                                }
                                nextService = Odometer + min;
                                pDate1 = p_day + "-" + p_month + "-" +
p_year;
                                if (min_month == 12) {
                                    p_year += 1;
                                } else if (min_month == 24) {
                                    p_year += 2;
                                } else if (min_month == 36) {
                                    p_year += 3;
                                } else if (min_month == 48) {
                                    p_year += 4;
                                }
                                p_month -= 1;
                                pDate = p_day + "-" + p_month + "-" +
p_year;
                                bool = true;


                            } catch (JSONException e) {
                                e.printStackTrace();
                            }
                            String stringBuilder1 = " \n";
                            tvHiddenTasks.setText(stringBuilder1);

                            for (int i = 0; i < visitScheduleList.size();
i++) {
                                int nextTasks = (int) (Odometer /
visitScheduleList.get(i).getKilometers());
                                nextTasks = Math.round(nextTasks);
//Rounded towards up if nextTasks==2,9 -> 2 rounded


                                if ((nextTasks == 1) || (nextTasks == 0)) {


                                    String url =
"http://192.168.1.3:8080/api/cartask/" +
visitScheduleList.get(i).getTaskCode();
                                    int finalI = i;
                                    JsonObjectRequest jsonObjectRequest =
new JsonObjectRequest(Request.Method.GET, url, null, new
Response.Listener<JSONObject>() {
                                        @Override
                                        public void onResponse(JSONObject
response) {

                                            if (((Odometer) ==
visitScheduleList.get(finalI).getKilometers())) {  //First service
```

```
                                                String taskPart =
response.optString("taskPart");
                                                String taskType =
response.optString("taskType");
                                                String tvText =
tvHiddenTasks.getText().toString().trim();
                                                String stringBuilder =
tvText + " \n" + taskPart + " (" + taskType + ")";

tvHiddenTasks.setText(stringBuilder);


                                            } else if
((visitScheduleList.get(finalI).getKilometers() - Odometer < 3001) &&
Odometer - visitScheduleList.get(finalI).getKilometers() > 0) {
                                                String taskPart =
response.optString("taskPart");
                                                String taskType =
response.optString("taskType");
                                                String tvText =
tvHiddenTasks.getText().toString().trim();
                                                String stringBuilder =
tvText + " \n" + taskPart + " (" + taskType + ")";

tvHiddenTasks.setText(stringBuilder);

                                            } else if
((visitScheduleList.get(finalI).getKilometers() - Odometer < 10001 &&
visitScheduleList.get(finalI).getKilometers() - Odometer > -5001) &&
Odometer - visitScheduleList.get(finalI).getKilometers() < 0) {
                                                String taskPart =
response.optString("taskPart");
                                                String taskType =
response.optString("taskType");
                                                String tvText =
tvHiddenTasks.getText().toString().trim();
                                                String stringBuilder =
tvText + " \n" + taskPart + " (" + taskType + ")";

tvHiddenTasks.setText(stringBuilder);


                                            }
                                        }
                                    }, new Response.ErrorListener() {
                                        @Override
                                        public void
onErrorResponse(VolleyError error) {
                                            String errormsg;
                                            errormsg =
VolleyErrorHelper.getMessage(error, MyCarsActivity.this);

Toast.makeText(MyCarsActivity.this, errormsg, Toast.LENGTH_SHORT).show();
                                        }
                                    });

RequestSingleton.getInstance(MyCarsActivity.this).addToRequestQueue(jsonObj
ectRequest);


                                } else if (nextTasks == 2) {
```

```
                                    String url =
"http://192.168.1.3:8080/api/cartask/" +
visitScheduleList.get(i).getTaskCode();
                                    JsonObjectRequest jsonObjectRequest =
new JsonObjectRequest(Request.Method.GET, url, null, new
Response.Listener<JSONObject>() {
                                        @Override
                                        public void onResponse(JSONObject
response) {
                                            String taskPart =
response.optString("taskPart");
                                            String taskType =
response.optString("taskType");
                                            String tvText =
tvHiddenTasks.getText().toString().trim();
                                            String stringBuilder = tvText +
" \n" + taskPart + " (" + taskType + ")";

tvHiddenTasks.setText(stringBuilder);


                                        }
                                    }, new Response.ErrorListener() {
                                        @Override
                                        public void
onErrorResponse(VolleyError error) {
                                            String errormsg;
                                            errormsg =
VolleyErrorHelper.getMessage(error, MyCarsActivity.this);

Toast.makeText(MyCarsActivity.this, errormsg, Toast.LENGTH_SHORT).show();
                                        }
                                    });

RequestSingleton.getInstance(MyCarsActivity.this).addToRequestQueue(jsonObj
ectRequest);


                                } else if (nextTasks == 4) {


                                    String url =
"http://192.168.1.3:8080/api/cartask/" +
visitScheduleList.get(i).getTaskCode();
                                    JsonObjectRequest jsonObjectRequest =
new JsonObjectRequest(Request.Method.GET, url, null, new
Response.Listener<JSONObject>() {
                                        @Override
                                        public void onResponse(JSONObject
response) {
                                            String taskPart =
response.optString("taskPart");
                                            String taskType =
response.optString("taskType");
                                            String tvText =
tvHiddenTasks.getText().toString().trim();
                                            String stringBuilder = tvText +
" \n" + taskPart + " (" + taskType + ")";

tvHiddenTasks.setText(stringBuilder);
```

```
                                        }
                                    }, new Response.ErrorListener() {
                                        @Override
                                        public void
onErrorResponse(VolleyError error) {
                                            String errormsg;
                                            errormsg =
VolleyErrorHelper.getMessage(error, MyCarsActivity.this);

Toast.makeText(MyCarsActivity.this, errormsg, Toast.LENGTH_SHORT).show();
                                        }
                                    });

RequestSingleton.getInstance(MyCarsActivity.this).addToRequestQueue(jsonObj
ectRequest);

                                } else if (nextTasks == 6) {


                                    String url =
"http://192.168.1.3:8080/api/cartask/" +
visitScheduleList.get(i).getTaskCode();
                                    JsonObjectRequest jsonObjectRequest =
new JsonObjectRequest(Request.Method.GET, url, null, new
Response.Listener<JSONObject>() {
                                        @Override
                                        public void onResponse(JSONObject
response) {
                                            String taskPart =
response.optString("taskPart");
                                            String taskType =
response.optString("taskType");
                                            String tvText =
tvHiddenTasks.getText().toString().trim();
                                            String stringBuilder = tvText +
" \n" + taskPart + " (" + taskType + ")";

tvHiddenTasks.setText(stringBuilder);


                                        }
                                    }, new Response.ErrorListener() {
                                        @Override
                                        public void
onErrorResponse(VolleyError error) {
                                            String errormsg;
                                            errormsg =
VolleyErrorHelper.getMessage(error, MyCarsActivity.this);

Toast.makeText(MyCarsActivity.this, errormsg, Toast.LENGTH_SHORT).show();
                                        }
                                    });

RequestSingleton.getInstance(MyCarsActivity.this).addToRequestQueue(jsonObj
ectRequest);

                                }
                            }
                            String tvText =
```

```
tvHiddenTasks.getText().toString().trim();
                            String stringBuilder = tvText + "\n:";
                            tvHiddenTasks.setText(stringBuilder);


                        }
                    }, new Response.ErrorListener() {
                        @Override
                        public void onErrorResponse(VolleyError error) {

                            String errormsg;
                            errormsg = VolleyErrorHelper.getMessage(error,
MyCarsActivity.this);
                            Toast.makeText(MyCarsActivity.this, errormsg,
Toast.LENGTH_SHORT).show();

                        }
                    });


                    //Add the json request object to the request queue

RequestSingleton.getInstance(MyCarsActivity.this).addToRequestQueue(jsonArr
ayRequest);




                    for (int codeW = 1; codeW < 8; codeW++) {
                        String url1 =
"http://192.168.1.3:8080/api/workshop/" + codeW;
                        JsonObjectRequest jsonObjectRequest1 = new
JsonObjectRequest(Request.Method.GET, url1, null, new
Response.Listener<JSONObject>() {
                            @Override
                            public void onResponse(JSONObject response) {

                                String name = response.optString("name");
                                String tvText =
tvHiddenWS.getText().toString().trim();
                                String stringBuilder = tvText + " \n" +
name;
                                tvHiddenWS.setText(stringBuilder);


                            }
                        }, new Response.ErrorListener() {
                            @Override
                            public void onErrorResponse(VolleyError error)
{
                                String errormsg;
                                errormsg =
VolleyErrorHelper.getMessage(error, MyCarsActivity.this);
                                Toast.makeText(MyCarsActivity.this,
errormsg, Toast.LENGTH_SHORT).show();
                            }
                        });

RequestSingleton.getInstance(MyCarsActivity.this).addToRequestQueue(jsonObj
ectRequest1);
```

```
                    }

                    Toast.makeText(MyCarsActivity.this,
getString(R.string.ypu_will_be_notified), Toast.LENGTH_SHORT).show();

                }
            }
        });



        myCarSave.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {



                String Brand =
myCarBrand.getSelectedItem().toString().trim(); //For the id repeat the
above requests
                String Model =
myCarModel.getSelectedItem().toString().trim(); //May 2 methods that get
those strings and return the ids.
                String Year =
myCarYear.getSelectedItem().toString().trim();
                String Cubic = myCarCubic.getText().toString().trim();
                String HorsePower =
myCarHorsePower.getText().toString().trim();
                String FuelType =
myCarFuelType.getText().toString().trim();
                String DateDay = myCarDateDay.getText().toString().trim();
                String DateMonth =
myCarDateMonth.getText().toString().trim();
                String DateYear =
myCarDateYear.getText().toString().trim();
                String Plate = myCarPlate.getText().toString().trim();
                String Chassis = myCarChassis.getText().toString().trim();
                String Color = myCarColor.getText().toString().trim();







                if (Brand.isEmpty() || DateDay.isEmpty() ||
DateMonth.isEmpty() || DateYear.isEmpty() || Plate.isEmpty() ||
Chassis.isEmpty() || Color.isEmpty()) {
                    Toast.makeText(getApplicationContext(),
getString(R.string.car_all_fields), Toast.LENGTH_SHORT).show();
                    //shows the toast if input field is empty
                } else if((!DateYear.matches("\\d+(?:\\.\\d+)?")) ||
(!DateDay.matches("\\d+(?:\\.\\d+)?")) ||
(!DateMonth.matches("\\d+(?:\\.\\d+)?")) ) {

                    Toast.makeText(getApplicationContext(),
getString(R.string.Wrong_data_input_msg_mycars),
Toast.LENGTH_SHORT).show();

                } else {
                String result = new
dbCarManager(MyCarsActivity.this).addCar(Brand, Model, Year, Cubic,
```

```
HorsePower, FuelType,DateDay, DateMonth, DateYear, Plate, Chassis, Color,
ModelCode);




                        Toast.makeText(getApplicationContext(),
getString(R.string.car_saved), Toast.LENGTH_SHORT).show();
                        //inserts the title,date,time into sql lite
database

                    if (bool) {
                        SharedPreferences n =
getSharedPreferences("Notification_prf", Context.MODE_PRIVATE);
                        notification = n.getBoolean("notif", true);
                        String stringTasks =
tvHiddenTasks.getText().toString().trim();
                        String stringWS =
tvHiddenWS.getText().toString().trim();

                        String title = getString(R.string.cv_rem_text) + "
" + Brand + " " + Model + " " + getString(R.string.with_chassis_number)
                                + " " + Chassis + " " +
getString(R.string.cv_rem_part2) + " " + stringTasks + " \n\n" +
getString(R.string.ws_text_rem) + "\n" + stringWS;
                        String time = "10:00AM";

                        String result1 = new
dbManager(MyCarsActivity.this).addreminder(title, pDate, time);
                        //inserts the title,date,time into sql lite
database
                        if (notification) {
                            setAlarm(title, pDate, time);
                            //calls the set alarm method to set alarm
                        } else {
                            Toast.makeText(getApplicationContext(),
getString(R.string.preference_msg_disabled_notif),
Toast.LENGTH_SHORT).show();
                        }

                    }

                        Intent intentBack = new
Intent(getApplicationContext(), MainActivity.class);
                        //this intent will be called once the setting alarm
completes
                        intentBack.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK |
Intent.FLAG_ACTIVITY_CLEAR_TASK);
                        startActivity(intentBack);
                    }




            }
        });
```

```
    }


    @Override
    public void onItemSelected(AdapterView<?> adapterView, View view, int
i, long l) {


        if(adapterView.getId() == R.id.mycar_brand){

            selectedBrand = adapterView.getSelectedItem().toString();
            String url =
"http://192.168.1.3:8080/api/byname/"+selectedBrand; //Finds the brandCode
of the brand name

            JsonObjectRequest jsonObjectRequest = new
JsonObjectRequest(Request.Method.GET, url, null, new
Response.Listener<JSONObject>() {
                @Override
                public void onResponse(JSONObject response) {
                    try {
                        JSONObject jsonObject = new
JSONObject(response.toString());
                        Long getBrandCode = jsonObject.optLong("id");
//The brandCode retrieved by the name of the brand
                        String url1
="http://192.168.1.3:8080/api/bybrand?brand="+ getBrandCode; //Finds the
car model by brandCode

                        JsonArrayRequest jsonArrayRequest = new
JsonArrayRequest(Request.Method.GET, url1, null, new
Response.Listener<JSONArray>() {
                    @Override
                    public void onResponse(JSONArray response) {
                    try {
                        JSONArray jsonArray = new
JSONArray(response.toString());
                        for(int i=0; i<jsonArray.length(); i++){
                            JSONObject jsonObject =
jsonArray.getJSONObject(i);
                            String model = jsonObject.optString("model");
                            if(i==0) {
                                modelsList.add(model); //First item on the
arraylist

                            } else {
                                if
(!model.equals(modelsList.get(modelsList.size()-1))) {
                                    modelsList.add(model); //The rest items
on the arraylist

                                }
                            }
                            year = jsonObject.optString("year");
                            yearList.add(year);

                            modelsAdapter = new
ArrayAdapter<>(MyCarsActivity.this,
                                    android.R.layout.simple_spinner_item,
modelsList);
```

```
modelsAdapter.setDropDownViewResource(android.R.layout.simple_spinner_dropd
own_item);
                            myCarModel.setAdapter(modelsAdapter);


                            yearAdapter = new
ArrayAdapter<>(MyCarsActivity.this,
                                    android.R.layout.simple_spinner_item,
yearList);

yearAdapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdow
n_item);
                            myCarYear.setAdapter(yearAdapter);
                        }
                    } catch (JSONException e) {
                        e.printStackTrace();
                    }
                    }
                }, new Response.ErrorListener() {
                    @Override
                    public void onErrorResponse(VolleyError error) {

                        String errormsg;
                        errormsg=
VolleyErrorHelper.getMessage(error,MyCarsActivity.this);
                        Toast.makeText(getApplicationContext(), errormsg,
Toast.LENGTH_SHORT).show();

                    }
                });


RequestSingleton.getInstance(MyCarsActivity.this).addToRequestQueue(jsonArr
ayRequest);
                        modelsList.clear();


myCarModel.setOnItemSelectedListener(MyCarsActivity.this);

                        //Get modelCode for the right year
                        yearList.clear();

myCarYear.setOnItemSelectedListener(MyCarsActivity.this);


                    } catch (JSONException e) {
                        e.printStackTrace();
                    }
                }
            }, new Response.ErrorListener() {
                @Override
                public void onErrorResponse(VolleyError error) {
                    String errormsg;
                    errormsg=
VolleyErrorHelper.getMessage(error,MyCarsActivity.this);
                    Toast.makeText(getApplicationContext(), errormsg,
Toast.LENGTH_SHORT).show();
                }
            });

RequestSingleton.getInstance(this).addToRequestQueue(jsonObjectRequest);
```

```
        } else if (adapterView.getId() == R.id.mycar_model) {

            modelName = adapterView.getSelectedItem().toString();


        }else if (adapterView.getId() == R.id.mycar_year){
            String year = adapterView.getSelectedItem().toString();
            String url = "http://192.168.1.3:8080/api/bymodelyear/"+
modelName +"/"+ year;
            JsonObjectRequest jsonObjectRequest = new
JsonObjectRequest(Request.Method.GET, url, null, new
Response.Listener<JSONObject>() {
                @Override
                public void onResponse(JSONObject response) {
                    try {
                        JSONObject modelInfo = new
JSONObject(response.toString());
                        String cubic = modelInfo.optString("cubic");
                        String horsePower =
modelInfo.optString("horsePower");
                        String fuelType = modelInfo.optString("fuelType");
                        ModelCode = modelInfo.optLong("id");

                        myCarCubic.setText(cubic);
                        myCarHorsePower.setText(horsePower);
                       myCarFuelType.setText(fuelType);

myCarModelCodeInvisible.setText(String.valueOf(ModelCode));

                    } catch (JSONException e) {
                        e.printStackTrace();
                    }

                }
            }, new Response.ErrorListener() {
                @Override
                public void onErrorResponse(VolleyError error) {

                    String errormsg;
                    errormsg=
VolleyErrorHelper.getMessage(error,MyCarsActivity.this);
                    Toast.makeText(getApplicationContext(), errormsg,
Toast.LENGTH_SHORT).show();

                }
            });


RequestSingleton.getInstance(this).addToRequestQueue(jsonObjectRequest);

        }

    }

    @Override
    public void onNothingSelected(AdapterView<?> adapterView) {

    }
```

```
    void setAlarm (String title,String date, String time) {
        //inserts the title,date,time into sql lite database

        AlarmManager am = (AlarmManager)
getSystemService(Context.ALARM_SERVICE);
        //assigining alaram manager object to set alaram

        Intent intent = new Intent(getApplicationContext(),
AlarmBroadcast.class);
        intent.putExtra("event", title);
        //sending data to alarm class to create channel and notification
        intent.putExtra("time", date);
        intent.putExtra("date", time);

        PendingIntent pendingIntent =
PendingIntent.getBroadcast(getApplicationContext(), 0, intent,
PendingIntent.FLAG_ONE_SHOT);
        String dateandtime = date + " " + time;
        DateFormat formatter = new SimpleDateFormat("d-M-yyyy hh:mm");
        try {
            Date date1 = formatter.parse(dateandtime);
            am.set(AlarmManager.RTC_WAKEUP, date1.getTime(),
pendingIntent);
            Toast.makeText(getApplicationContext(),
getString(R.string.rem_set_success_msg), Toast.LENGTH_SHORT).show();

        } catch (ParseException e) {
            e.printStackTrace();
        }
    }
}
```

MyCarsFragment.java

```
package com.example.carbook.mycars;

import android.content.Intent;
import android.database.Cursor;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;
import android.widget.Toast;

import androidx.annotation.Nullable;
import androidx.appcompat.app.AppCompatActivity;
import androidx.fragment.app.Fragment;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

import com.example.carbook.R;
import
com.google.android.material.floatingactionbutton.FloatingActionButton;

import java.util.ArrayList;


public class MyCarsFragment extends Fragment {
```

```
    FloatingActionButton mCreateCar;
    RecyclerView myCarRecyclerview;
    ArrayList<MyCarModel> dataholder = new ArrayList<MyCarModel>();
    myCarAdapter adapter;

    TextView tvEmpty;

    @Nullable
    @Override
    public View onCreateView(LayoutInflater inflater, @Nullable  ViewGroup
container, @Nullable  Bundle savedInstanceState) {

        View view = inflater.inflate(R.layout.fragment_mycars, container,
false);


((AppCompatActivity)getActivity()).getSupportActionBar().setTitle(getString
(R.string.my_cars));

        tvEmpty = view.findViewById(R.id.empty);

        myCarRecyclerview = view.findViewById(R.id.car_recycler);
        myCarRecyclerview.setLayoutManager(new
LinearLayoutManager(getContext()));
        mCreateCar = view.findViewById(R.id.add_car);
        //Floating action button to change activity
        mCreateCar.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Intent intent = new Intent(getContext(),
MyCarsActivity.class);
                startActivity(intent);
                //Starts the new activity to add Cars
            }
        });

        Cursor cursor1 = new dbCarManager(getContext()).carIsEmpty();
        cursor1.moveToFirst();
        int count = cursor1.getInt(0);
        if(count != 0) {
            tvEmpty.setVisibility(View.INVISIBLE);
        } else {tvEmpty.setVisibility(View.VISIBLE);}

        Cursor cursor = new dbCarManager(getContext()).readallcars();
//Cursor To Load data From the database
        while (cursor.moveToNext()) {
            MyCarModel model = new
MyCarModel(cursor.getLong(0),cursor.getString(1), cursor.getString(2),
cursor.getString(3),
                    cursor.getString(4), cursor.getString(5),
cursor.getString(6), cursor.getString(7),
                    cursor.getString(8), cursor.getString(9),
cursor.getString(10), cursor.getString(11), cursor.getString(12),
cursor.getLong(13));
            dataholder.add(model);
        }

        adapter = new myCarAdapter(getActivity(),getContext(),dataholder);
        myCarRecyclerview.setAdapter(adapter); //Binds the adapter with
recyclerview
```

```
        myCarRecyclerview.setLayoutManager(new
LinearLayoutManager(getActivity()));

        return view;


    }
}
```

MyCarsViewActivity.java

```
package com.example.carbook.mycars;

import android.app.AlertDialog;
import android.content.DialogInterface;
import android.content.Intent;
import android.database.Cursor;
import android.os.Bundle;
import android.view.View;
import android.widget.TextView;
import android.widget.Toast;

import androidx.appcompat.app.ActionBar;
import androidx.appcompat.app.AppCompatActivity;

import com.example.carbook.R;
import com.example.carbook.mainactivity.MainActivity;
import
com.google.android.material.floatingactionbutton.FloatingActionButton;

public class MyCarsViewActivity extends AppCompatActivity  {
    TextView myCarCubic, myCarHorsePower,myCarFuelType,myCarBrand,
myCarModel, myCarYear, myCarDateDay, myCarDateMonth, myCarDateYear,
myCarPlate, myCarChassis, myCarColor,myCarModelCodeInvisible;
    FloatingActionButton myCarDelete;
    String model, brand,dateday,datemonth, dateyear, chassis, plate, color,
year,cubic,horsepower,fueltype;
    Long modelcode;



    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_my_cars_view);
        ActionBar titleS = getSupportActionBar();
        titleS.hide();


        //access the data form the input fields and spinners
        myCarBrand = findViewById(R.id.mycar_brand1);
        myCarModel = findViewById(R.id.mycar_model1);
        myCarYear = findViewById(R.id.mycar_year1);

        myCarCubic = findViewById(R.id.mycar_cubic1);
        myCarHorsePower = findViewById(R.id.mycar_horsepower1);
        myCarFuelType = findViewById(R.id.mycar_fueltype1);
```

```
        myCarDateDay = findViewById(R.id.mycar_date_day);
        myCarDateMonth = findViewById(R.id.mycar_date_month);
        myCarDateYear = findViewById(R.id.mycar_date1);
        myCarPlate = findViewById(R.id.mycar_plate1);
        myCarChassis = findViewById(R.id.mycar_chassis1);
        myCarColor = findViewById(R.id.mycar_color1);

        myCarCubic = findViewById(R.id.mycar_cubic1);
        myCarHorsePower = findViewById(R.id.mycar_horsepower1);
        myCarFuelType = findViewById(R.id.mycar_fueltype1);
        myCarModelCodeInvisible = findViewById(R.id.invisible_model_code1);

        myCarDelete = findViewById(R.id.mycar_delete);

        getSetIntentData();

        myCarDelete.setOnClickListener(new View.OnClickListener() {


            @Override
            public void onClick(View view) {
                confirmDialog();
            }
        });

    }

    void getSetIntentData(){
        if(getIntent().hasExtra("model")){
            //Getting data from intent
            brand = getIntent().getStringExtra("brand");
            model = getIntent().getStringExtra("model");
            year = getIntent().getStringExtra("year");
            cubic = getIntent().getStringExtra("cubic");
            horsepower = getIntent().getStringExtra("horsepower");
            fueltype = getIntent().getStringExtra("fueltype");
            dateday = getIntent().getStringExtra("dateday");
            datemonth = getIntent().getStringExtra("datemonth");
            dateyear = getIntent().getStringExtra("dateyear");
            chassis = getIntent().getStringExtra("chassis");
            plate = getIntent().getStringExtra("plate");
            color = getIntent().getStringExtra("color");
            modelcode = getIntent().getLongExtra("modelcode",0);


            myCarBrand.setText(brand);
            myCarModel.setText(model);
            myCarYear.setText(year);
            myCarCubic.setText(cubic);
            myCarHorsePower.setText(horsepower);
            myCarFuelType.setText(fueltype);
            myCarDateDay.setText(dateday);
            myCarDateMonth.setText(datemonth);
            myCarDateYear.setText(dateyear);
            myCarChassis.setText(chassis);
            myCarPlate.setText(plate);
            myCarColor.setText(color);

myCarModelCodeInvisible.setText(String.valueOf(Math.toIntExact(modelcode)))
;
```

```
        }


    }

    void confirmDialog(){
        AlertDialog.Builder builder = new AlertDialog.Builder(this);
        builder.setMessage(R.string.mycar_delete_affirmation);
        builder.setPositiveButton(R.string.btn_delete_remdialog, new
DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialogInterface, int i) {
                String DateYear =
myCarDateYear.getText().toString().trim();
                Cursor cursor = new
dbCarManager(MyCarsViewActivity.this).getMyCarId(DateYear);
                cursor.moveToFirst();
                long car_id = cursor.getLong(0);
                String result1 = new
dbCarManager(MyCarsViewActivity.this).deleteVisit(car_id);
                String result = new
dbCarManager(MyCarsViewActivity.this).deletecar(car_id);
                Toast.makeText(getApplicationContext(),
getString(R.string.car_deleted), Toast.LENGTH_SHORT).show();
                finish();
                Intent intentBack = new Intent(getApplicationContext(),
MainActivity.class);
                //this intent will be called once the setting alarm
completes
                intentBack.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK |
Intent.FLAG_ACTIVITY_CLEAR_TASK);
                startActivity(intentBack);
            }
        });
        builder.setNegativeButton(R.string.btn_cancel_remdialog, new
DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialogInterface, int i) {
                Toast.makeText(getApplicationContext(),
getString(R.string.del_cancel_remdialog), Toast.LENGTH_SHORT).show();
            }
        });
        builder.create().show();
    }


}
```

VolleyErrorHelper.java

```
package com.example.carbook.mycars;

import android.content.Context;

import com.android.volley.AuthFailureError;
import com.android.volley.NetworkError;
import com.android.volley.NetworkResponse;
import com.android.volley.NoConnectionError;
import com.android.volley.ServerError;
import com.android.volley.TimeoutError;
```

```
import com.android.volley.VolleyError;
import com.example.carbook.R;
import com.google.gson.Gson;
import com.google.gson.reflect.TypeToken;

import java.util.HashMap;
import java.util.Map;

public class VolleyErrorHelper {
    /**
     * Returns appropriate message which is to be displayed to the user
     * against the specified error object.
     *
     * @param error
     * @param context
     * @return
     */
    public static String getMessage(Object error, Context context) {
        if (error instanceof TimeoutError) {
            return
context.getResources().getString(R.string.generic_server_down);
        }
        else if (isServerProblem(error)) {
            return handleServerError(error, context);
        }
        else if (isNetworkProblem(error)) {
            return context.getResources().getString(R.string.no_internet);
        }
        return context.getResources().getString(R.string.generic_error);
    }

    /**
     * Determines whether the error is related to network
     * @param error
     * @return
     */
    private static boolean isNetworkProblem(Object error) {
        return (error instanceof NetworkError) || (error instanceof
NoConnectionError);
    }
    /**
     * Determines whether the error is related to server
     * @param error
     * @return
     */
    private static boolean isServerProblem(Object error) {
        return (error instanceof ServerError) || (error instanceof
AuthFailureError);
    }
    /**
     * Handles the server error, tries to determine whether to show a stock
message or to
     * show a message retrieved from the server.
     *
     * @param err
     * @param context
     * @return
     */
    private static String handleServerError(Object err, Context context) {
        VolleyError error = (VolleyError) err;
```

```
        NetworkResponse response = error.networkResponse;

        if (response != null) {
            switch (response.statusCode) {
                case 404:
                case 422:
                case 401:
                    try {
                        // server might return error like this { "error":
"Some error occured" }
                        // Use "Gson" to parse the result
                        HashMap<String, String> result = new
Gson().fromJson(new String(response.data),
                                new TypeToken<Map<String, String>>() {
                                }.getType());

                        if (result != null && result.containsKey("error"))
{
                            return result.get("error");
                        }

                    } catch (Exception e) {
                        e.printStackTrace();
                    }
                    // invalid request
                    return error.getMessage();

                default:
                    return
context.getResources().getString(R.string.generic_server_down);
            }
        }
        return context.getResources().getString(R.string.generic_error);
    }
}
```

MyWorkshopsFragment.java

```
package com.example.carbook.myworkshops;

import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Toast;

import androidx.annotation.Nullable;
import androidx.appcompat.app.AppCompatActivity;
import androidx.fragment.app.Fragment;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

import com.android.volley.Request;
import com.android.volley.Response;
import com.android.volley.VolleyError;
import com.android.volley.toolbox.JsonArrayRequest;
import com.android.volley.toolbox.JsonObjectRequest;
import com.example.carbook.R;
```

```
import com.example.carbook.mainactivity.RequestSingleton;
import com.example.carbook.mycars.VolleyErrorHelper;
import com.example.carbook.mycars.dbCarManager;

import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

import java.util.ArrayList;

public class MyWorkshopsFragment extends Fragment {

    RecyclerView workshopsRecyclerview;
    ArrayList<WorkshopsModel> dataholder = new ArrayList<WorkshopsModel>();
    ArrayList<Long> AuthCodeW = new ArrayList<Long>();
    WorkshopsAdapter workshopsAdapter;

    @Nullable
    @Override
    public View onCreateView(LayoutInflater inflater, @Nullable  ViewGroup
container, @Nullable  Bundle savedInstanceState) {

        View view = inflater.inflate(R.layout.fragment_myworkshops,
container, false );


((AppCompatActivity)getActivity()).getSupportActionBar().setTitle(getString
(R.string.my_workshops));


        workshopsRecyclerview = view.findViewById(R.id.ws_recycler);
        workshopsRecyclerview.setLayoutManager(new
LinearLayoutManager(getContext()));

        Cursor cursor1 = new dbCarManager(getContext()).carIsEmpty();
        cursor1.moveToFirst();
        int count = cursor1.getInt(0);
        if(count == 0) {
            Toast.makeText(getContext(),
getString(R.string.ws_msg_workshops), Toast.LENGTH_SHORT).show();
        } else {

            String url = "http://192.168.1.3:8080/api/workshops"; //For
android phone

            JsonArrayRequest jsonArrayRequest = new
JsonArrayRequest(Request.Method.GET, url, null, new
Response.Listener<JSONArray>() {
                @Override
                public void onResponse(JSONArray response) {

                    try {
                        JSONArray jsonArray = new
JSONArray(response.toString());
                        for (int i = 0; i < jsonArray.length(); i++) {
                            JSONObject jsonObject =
jsonArray.getJSONObject(i);
                            Long code = jsonObject.optLong("id");
                            String name = jsonObject.optString("name");
                            String area = jsonObject.optString("area");
                            String address =
```

```
jsonObject.optString("address");
                            String coordinates =
jsonObject.optString("coordinates");
                            String telephone =
jsonObject.optString("telephone");


                            dataholder.add(new WorkshopsModel(code, name,
area, address, coordinates, telephone));


                        }
                        workshopsAdapter = new
WorkshopsAdapter(getActivity(), getContext(), dataholder);
                        workshopsRecyclerview.setAdapter(workshopsAdapter);
//Binds the adapter with recyclerview

                    } catch (JSONException e) {
                        e.printStackTrace();
                    }

                }
            }, new Response.ErrorListener() {
                @Override
                public void onErrorResponse(VolleyError error) {

                    String errormsg;
                    errormsg = VolleyErrorHelper.getMessage(error,
getContext());
                    Toast.makeText(getContext(), errormsg,
Toast.LENGTH_SHORT).show();

                }
            });

            //Add the json request object to the request queue

RequestSingleton.getInstance(getContext()).addToRequestQueue(jsonArrayReque
st);

        }



        //Show workshops according to my car brands

                                Cursor cursor = new
dbCarManager(getContext()).getMyCarModel1();                    //Cursor To
Load data From the database
                                while (cursor.moveToNext()) {
                                    String brand = cursor.getString(0);
                                    String url11 =
"http://192.168.1.3:8080/api/byname/"+brand;
                                    JsonObjectRequest jsonObjectRequest =
new JsonObjectRequest(Request.Method.GET, url11, null, new
Response.Listener<JSONObject>() {
                                        @Override
                                        public void onResponse(JSONObject
response) {
                                            //JSONObject jsonObject = new
JSONObject(response.toString());
```

```
                                                Long id =
response.optLong("id");
                                                String url1 =
"http://192.168.1.3:8080/api/authorizedworkshops?brands="+id;
                                                //JsonArrayRequest
                                                JsonArrayRequest
jsonArrayRequest = new JsonArrayRequest(Request.Method.GET, url1, null, new
Response.Listener<JSONArray>() {
                                                    @Override
                                                    public void
onResponse(JSONArray response) {
                                                        JSONArray jsonArray =
null;
                                                        try {
                                                            jsonArray = new
JSONArray(response.toString());
                                                            for(int i=0;
i<jsonArray.length(); i++) {
                                                                JSONObject
jsonObject = jsonArray.getJSONObject(i);
                                                                Long codeW =
jsonObject.optLong("codeW");
                                                                Long brands =
jsonObject.optLong("brands");
                                                                if
(brands.equals(id)){
                                                                   String url3
= "http://192.168.1.3:8080/api/workshop/"+id; //codeW

JsonObjectRequest jsonObjectRequest1 = new
JsonObjectRequest(Request.Method.GET, url3, null, new
Response.Listener<JSONObject>() {

@Override
                                                                        public
void onResponse(JSONObject response) {

Long code = response.optLong("id");

String name = response.optString("name");

String area = response.optString("area");

String address = response.optString("address");

String coordinates = response.optString("coordinates");

String telephone = response.optString("telephone");


for(int i=0;i<dataholder.size();i++){

if(!(dataholder.get(i).getCodeW().equals(code))){

dataholder.add(new
WorkshopsModel(code,name,area,address,coordinates,telephone));

}
                                                                        }

                                                                    }
```

```
                                                                    }, new
Response.ErrorListener() {

@Override
                                                                        public
void onErrorResponse(VolleyError error) {

                                                                        }
                                                                    });

RequestSingleton.getInstance(getContext()).addToRequestQueue(jsonObjectRequ
est1);
                                                                }

                                                            }

                                                        } catch (JSONException
e) {

e.printStackTrace();
                                                        }

                                                    }
                                                }, new Response.ErrorListener()
{
                                                    @Override
                                                    public void
onErrorResponse(VolleyError error) {
                                                        String errormsg;
                                                        errormsg=
VolleyErrorHelper.getMessage(error,getContext());

Toast.makeText(getContext(), errormsg, Toast.LENGTH_SHORT).show();
                                                    }
                                                });

RequestSingleton.getInstance(getContext()).addToRequestQueue(jsonArrayReque
st);

                                            }
                                        }, new Response.ErrorListener() {
                                            @Override
                                            public void
onErrorResponse(VolleyError error) {
                                                String errormsg;
                                                errormsg=
VolleyErrorHelper.getMessage(error,getContext());
                                                Toast.makeText(getContext(),
errormsg, Toast.LENGTH_SHORT).show();
                                            }
                                        });

RequestSingleton.getInstance(getContext()).addToRequestQueue(jsonObjectRequ
est);
                                    }
```

```
        return view;

    }
}
```

MyWorkshopActivity.java

```
package com.example.carbook.myworkshops;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.os.Bundle;
import android.view.MotionEvent;
import android.view.View;
import android.widget.ArrayAdapter;
import android.widget.ListView;
import android.widget.TextView;
import android.widget.Toast;

import com.android.volley.Request;
import com.android.volley.Response;
import com.android.volley.VolleyError;
import com.android.volley.toolbox.JsonArrayRequest;
import com.android.volley.toolbox.JsonObjectRequest;
import com.example.carbook.R;
import com.example.carbook.mainactivity.MainActivity;
import com.example.carbook.mainactivity.RequestSingleton;
import com.example.carbook.mycars.VolleyErrorHelper;
import
com.google.android.material.floatingactionbutton.FloatingActionButton;

import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

import java.util.ArrayList;

public class WorkshopActivity extends AppCompatActivity {

    TextView name, address, area, telephone;
    ListView wsTypeList,wsBrandsList,wsAuthorizedList;
    FloatingActionButton showMap;
    String ws_name, ws_address, ws_area, ws_telephone, ws_coordinates;
    Long ws_codeW;

    ArrayList<String> typeList = new ArrayList<>();
    ArrayList<String> authBrands = new ArrayList<>();
    ArrayList<String> authOrNot = new ArrayList<>();

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_workshop);

        name = findViewById(R.id.ws_name);
        address = findViewById(R.id.ws_address);
        area = findViewById(R.id.ws_city);
        telephone = findViewById(R.id.ws_telephone);
```

```
        wsTypeList = findViewById(R.id.lv_typeWS);
        wsBrandsList = findViewById(R.id.lv_brandsWS);


        wsAuthorizedList = findViewById(R.id.lv_authorizedWS);

        showMap = findViewById(R.id.ws_showmap);

        //Setting the information of the workshop
        getSetIntentData();

        wsTypeList.setOnTouchListener(new ListView.OnTouchListener() {
            @Override
            public boolean onTouch(View v, MotionEvent event) {
                int action = event.getAction();
                switch (action) {
                    case MotionEvent.ACTION_DOWN:
                        // Disallow ScrollView to intercept touch events.
v.getParent().requestDisallowInterceptTouchEvent(true);
                        break;

                    case MotionEvent.ACTION_UP:
                        // Allow ScrollView to intercept touch events.
v.getParent().requestDisallowInterceptTouchEvent(false);
                        break;
                }

                // Handle ListView touch events.
                v.onTouchEvent(event);
                return true;
            }
        });

        showMap.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent(getApplicationContext(),
WorkshopMapsActivity.class);
                intent.putExtra("coordinates", ws_coordinates);
                intent.putExtra("name", ws_name);
                startActivity(intent);

            }
        });

        //Show the workshop types

        String url = "http://192.168.1.3:8080/api/wstype?codeW=" + ws_codeW
; //For android phone

        JsonArrayRequest jsonArrayRequest = new
JsonArrayRequest(Request.Method.GET, url, null, new
Response.Listener<JSONArray>() {
            @Override
            public void onResponse(JSONArray response) {

                try {
                    JSONArray jsonArray = new
JSONArray(response.toString());
```

```
                    for(int i=0; i<jsonArray.length(); i++){
                        JSONObject jsonObject = jsonArray.getJSONObject(i);
                            JSONObject jsonObject1 =
jsonObject.getJSONObject("wst");
                            String taskType =
jsonObject1.optString("taskType");
                            typeList.add(taskType);



                    }
                    ArrayAdapter<String> adapter = new
ArrayAdapter<>(WorkshopActivity.this,
android.R.layout.simple_list_item_1,typeList);
                    wsTypeList.setAdapter(adapter);


                } catch (JSONException e) {
                    e.printStackTrace();
                }

            }
        }, new Response.ErrorListener() {
            @Override
            public void onErrorResponse(VolleyError error) {

                String errormsg;
                errormsg=
VolleyErrorHelper.getMessage(error,WorkshopActivity.this);
                Toast.makeText(WorkshopActivity.this, errormsg,
Toast.LENGTH_SHORT).show();

            }
        });

        //Add the json request object to the request queue

RequestSingleton.getInstance(WorkshopActivity.this).addToRequestQueue(jsonA
rrayRequest);




        //Show the authorized brand list
        String url1 =
"http://192.168.1.3:8080/api/authorizedbrands?codeW="+ ws_codeW; //For
android phone

        JsonArrayRequest jsonArrayRequest1 = new
JsonArrayRequest(Request.Method.GET, url1, null, new
Response.Listener<JSONArray>() {
            @Override
            public void onResponse(JSONArray response) {

                try {
                    JSONArray jsonArray = new
JSONArray(response.toString());
                    for(int i=0; i<jsonArray.length(); i++){
                        JSONObject jsonObject = jsonArray.getJSONObject(i);
                        Long brands = jsonObject.optLong("brands");
                        int authorizedBool =
```

```
jsonObject.optInt("authorized");
                        String authorized;
                        if(authorizedBool==1){

authorized=getString(R.string.authorized_ws_yes);
                        } else {

authorized=getString(R.string.authorized_ws_no);
                        }
                        authOrNot.add(authorized);

                        ArrayAdapter<String> adapter1 = new
ArrayAdapter<>(WorkshopActivity.this,
android.R.layout.simple_list_item_1,authOrNot);
                        wsAuthorizedList.setAdapter(adapter1);

                        String brandUrl =
"http://192.168.1.3:8080/api/carbrand/" +brands;
                        JsonObjectRequest singleJsonObject = new
JsonObjectRequest(Request.Method.GET, brandUrl, null, new
Response.Listener<JSONObject>() {
                            @Override
                            public void onResponse(JSONObject response) {
                                String brandName =
response.optString("name");
                                authBrands.add(brandName);

                                ArrayAdapter<String> adapter2 = new
ArrayAdapter<>(WorkshopActivity.this,
android.R.layout.simple_list_item_1,authBrands);
                                wsBrandsList.setAdapter(adapter2);

                            }
                        }, new Response.ErrorListener() {
                            @Override
                            public void onErrorResponse(VolleyError error)
{
                                String errormsg;
                                errormsg=
VolleyErrorHelper.getMessage(error,WorkshopActivity.this);
                                Toast.makeText(WorkshopActivity.this,
errormsg, Toast.LENGTH_SHORT).show();
                            }
                        });


RequestSingleton.getInstance(WorkshopActivity.this).addToRequestQueue(singl
eJsonObject);



                    }


                } catch (JSONException e) {
                    e.printStackTrace();
                }

            }
        }, new Response.ErrorListener() {
            @Override
```

```
            public void onErrorResponse(VolleyError error) {

                String errormsg;
                errormsg=
VolleyErrorHelper.getMessage(error,WorkshopActivity.this);
                Toast.makeText(WorkshopActivity.this, errormsg,
Toast.LENGTH_SHORT).show();

            }
        });

        //Add the json request object to the request queue

RequestSingleton.getInstance(WorkshopActivity.this).addToRequestQueue(jsonA
rrayRequest1);



    }

    void getSetIntentData() {
        if (getIntent().hasExtra("coordinates")) {
            //Getting data from intent
            ws_codeW = getIntent().getLongExtra("id",1);
            ws_name = getIntent().getStringExtra("name");
            ws_address = getIntent().getStringExtra("address");
            ws_area = getIntent().getStringExtra("area");
            ws_telephone = getIntent().getStringExtra("telephone");
            ws_coordinates = getIntent().getStringExtra("coordinates");


            name.setText(ws_name);
            address.setText(ws_address);
            area.setText(ws_area);
            telephone.setText(ws_telephone);

        }
    }

}
```

WorkshopMapsActivity.java

```
package com.example.carbook.myworkshops;

import android.os.Bundle;

import androidx.fragment.app.FragmentActivity;

import com.example.carbook.R;
import com.example.carbook.databinding.ActivityWorkshopMapsBinding;
import com.google.android.gms.maps.CameraUpdateFactory;
import com.google.android.gms.maps.GoogleMap;
import com.google.android.gms.maps.OnMapReadyCallback;
import com.google.android.gms.maps.SupportMapFragment;
import com.google.android.gms.maps.model.LatLng;
import com.google.android.gms.maps.model.MarkerOptions;

public class WorkshopMapsActivity extends FragmentActivity implements
OnMapReadyCallback {
```

```
    private GoogleMap mMap;
    private ActivityWorkshopMapsBinding binding;
    String ws_coordinates, ws_name;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);


        binding = ActivityWorkshopMapsBinding.inflate(getLayoutInflater());
        setContentView(binding.getRoot());

        // Obtain the SupportMapFragment and get notified when the map is
ready to be used.
        SupportMapFragment mapFragment = (SupportMapFragment)
getSupportFragmentManager()
                .findFragmentById(R.id.map);
        mapFragment.getMapAsync(this);

        if (getIntent().hasExtra("coordinates")) {
            ws_coordinates = getIntent().getStringExtra("coordinates");
            ws_name = getIntent().getStringExtra("name");
        }

    }

    /**
     * Manipulates the map once available.
     * This callback is triggered when the map is ready to be used.
     * This is where we can add markers or lines, add listeners or move the
camera. In this case,
     * we just add a marker near Sydney, Australia.
     * If Google Play services is not installed on the device, the user
will be prompted to install
     * it inside the SupportMapFragment. This method will only be triggered
once the user has
     * installed Google Play services and returned to the app.
     */
    @Override
    public void onMapReady(GoogleMap googleMap) {
        mMap = googleMap;

        mMap.getUiSettings().setZoomControlsEnabled(true);
        mMap.getUiSettings().setZoomGesturesEnabled(true);
        mMap.getUiSettings().setCompassEnabled(true);
        mMap.setTrafficEnabled(true);

        String[] latlong =  ws_coordinates.split(" ");
        double latitude = Double.parseDouble(latlong[0]);
        double longitude = Double.parseDouble(latlong[1]);
        // Add a marker in the location and move the camera
        LatLng location = new LatLng(latitude, longitude);
        mMap.addMarker(new
MarkerOptions().position(location).title(ws_name));
        mMap.animateCamera(CameraUpdateFactory.newLatLngZoom(location,
17));
    }
}
```

WorkshopsAdapter.java

```
package com.example.carbook.myworkshops;

import android.annotation.SuppressLint;
import android.app.Activity;
import android.content.Context;
import android.content.Intent;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;
import android.widget.Toast;

import androidx.annotation.NonNull;
import androidx.recyclerview.widget.RecyclerView;

import com.example.carbook.R;

import java.util.ArrayList;

public class WorkshopsAdapter extends
RecyclerView.Adapter<WorkshopsAdapter.myviewholder> {

    ArrayList<WorkshopsModel> WSdataholder;
    Context context;
    Activity activity;
    //array list to hold the

    public WorkshopsAdapter (Activity activity, Context context,
ArrayList<WorkshopsModel> WSdataholder) {
        this.WSdataholder = WSdataholder;
        this.context=context;
        this.activity=activity;
    }


    @NonNull
    @Override
    public WorkshopsAdapter.myviewholder onCreateViewHolder(@NonNull
ViewGroup parent, int viewType) {
        View view =
LayoutInflater.from(parent.getContext()).inflate(R.layout.single_workshop,
parent, false);
        LayoutInflater layoutInflater = LayoutInflater.from(context);


        //inflates the xml file in recyclerview
        return new WorkshopsAdapter.myviewholder(view);
    }

    @Override
    public void onBindViewHolder(@NonNull WorkshopsAdapter.myviewholder
holder, @SuppressLint("RecyclerView") int position) {

        WorkshopsModel currentItem = WSdataholder.get(position);
        String Name = currentItem.getName();
        String Area = currentItem.getArea();

        //Binds the single objects to recycler view
```

```
        holder.wName.setText(Name);
        holder.wArea.setText(Area);

        //  Toast.makeText(WorkshopsAdapter.this, ,
Toast.LENGTH_LONG).show();

        holder.itemView.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Intent intent = new Intent(context,
WorkshopActivity.class);
                intent.putExtra("id",
WSdataholder.get(position).getCodeW());
                intent.putExtra("name",
String.valueOf(WSdataholder.get(position).getName()));
                intent.putExtra("area",
String.valueOf(WSdataholder.get(position).getArea()));
                intent.putExtra("address",
String.valueOf(WSdataholder.get(position).getAddress()));
                intent.putExtra("telephone",
String.valueOf(WSdataholder.get(position).getTelephone()));
                intent.putExtra("coordinates",
String.valueOf(WSdataholder.get(position).getCoordinates()));
                //Starts the new activity
                activity.startActivity(intent);
            }
        });

    }

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

    class myviewholder extends RecyclerView.ViewHolder {

        TextView wName, wArea;


        public myviewholder(@NonNull View itemView) {
            super(itemView);

            wName = itemView.findViewById(R.id.tv_workshop);
//holds the reference of the materials to show data in recyclerview
            wArea = itemView.findViewById(R.id.tv_area);

        }
    }
}
```

WorkshopsModel.java

```
package com.example.carbook.myworkshops;

public class WorkshopsModel {

        Long codeW;
        String name, area, address, coordinates, telephone;
```

```
        public Long getCodeW() {
                return codeW;
        }

        public String getName() {
                return name;
        }

        public String getArea() {
                return area;
        }

        public String getAddress() {
                return address;
        }

        public String getCoordinates() {
                return coordinates;
        }

        public String getTelephone() {
                return telephone;
        }

        public WorkshopsModel (Long codeW, String name, String area, String
address, String coordinates, String telephone){
                this.codeW = codeW;
                this.name = name;
                this.area = area;
                this.address = address;
                this.coordinates =coordinates;
                this.telephone = telephone;
        }

}
```

AlarmBroadcast.java

```
package com.example.carbook.reminders;

import android.annotation.SuppressLint;
import android.app.Notification;
import android.app.NotificationChannel;
import android.app.NotificationManager;
import android.app.PendingIntent;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.os.Build;
import android.os.Bundle;
import android.widget.RemoteViews;

import androidx.core.app.NotificationCompat;

import com.example.carbook.R;

public class AlarmBroadcast extends BroadcastReceiver {
    @Override
    public void onReceive(Context context, Intent intent) {
        Bundle bundle = intent.getExtras();
```

```
        String text = bundle.getString("event");
        String date = bundle.getString("date") + " " +
bundle.getString("time");

        //Click on Notification
        Intent intent1 = new Intent(context, NotificationMessage.class);
        intent1.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
        intent1.putExtra("message", text);

        //Notification Builder
        PendingIntent pendingIntent = PendingIntent.getActivity(context, 1,
intent1, PendingIntent.FLAG_ONE_SHOT);
        NotificationManager notificationManager = (NotificationManager)
context.getSystemService(Context.NOTIFICATION_SERVICE);
        NotificationCompat.Builder mBuilder = new
NotificationCompat.Builder(context, "notify_001");


        //here we set all the properties for the notification
        @SuppressLint("RemoteViewLayout") RemoteViews contentView = new
RemoteViews(context.getPackageName(), R.layout.notification_layout);
        contentView.setImageViewResource(R.id.image, R.mipmap.ic_launcher);
        PendingIntent pendingSwitchIntent =
PendingIntent.getBroadcast(context, 0, intent, 0);
        //contentView.setOnClickPendingIntent(R.id.flashButton,
pendingSwitchIntent);
        contentView.setTextViewText(R.id.message, text);
        contentView.setTextViewText(R.id.date, date);
        mBuilder.setSmallIcon(R.drawable.alaram);
        mBuilder.setAutoCancel(true);
        mBuilder.setOngoing(true);
        mBuilder.setAutoCancel(true);
        mBuilder.setPriority(Notification.PRIORITY_HIGH);
        mBuilder.setOnlyAlertOnce(true);
        mBuilder.build().flags = Notification.FLAG_NO_CLEAR |
Notification.PRIORITY_HIGH;
        mBuilder.setContent(contentView);
        mBuilder.setContentIntent(pendingIntent);

        //we have to create notification channel after api level 26
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
            String channelId = "channel_id";
            NotificationChannel channel = new
NotificationChannel(channelId, "channel name",
NotificationManager.IMPORTANCE_HIGH);
            channel.enableVibration(true);
            notificationManager.createNotificationChannel(channel);
            mBuilder.setChannelId(channelId);
        }

        Notification notification = mBuilder.build();
        notificationManager.notify(1, notification);


    }
}
```

dbManager.java

```
package com.example.carbook.reminders;

import android.content.ContentValues;
import android.content.Context;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;

import androidx.annotation.Nullable;

public class dbManager extends SQLiteOpenHelper {
    private static String dbname = "reminder";
//Table  name to store reminders in sqllite

    public dbManager(@Nullable Context context) {
        super(context, dbname, null, 1);
    }

    @Override
    public void onCreate(SQLiteDatabase sqLiteDatabase) {
//sql query to insert data in sqllite
        String query = "create table tbl_reminder(id integer primary key
autoincrement,title text,date text,time text)";
        sqLiteDatabase.execSQL(query);
    }

    @Override
    public void onUpgrade(SQLiteDatabase sqLiteDatabase, int i, int i1) {

        String query = "DROP TABLE IF EXISTS tbl_reminder";
//sql query to check table with the same name or not
        sqLiteDatabase.execSQL(query);
//executes the sql command
        onCreate(sqLiteDatabase);

    }

    public Cursor reminderIsEmpty() {
        SQLiteDatabase database = this.getWritableDatabase();
        String query = "select count(*) from tbl_reminder";
        Cursor cursor = database.rawQuery(query, null);
        return cursor;
    }

    public String addreminder(String title, String date, String time) {
        SQLiteDatabase database = this.getReadableDatabase();

        ContentValues contentValues = new ContentValues();
        contentValues.put("title", title);
//Inserts  data into sqllite database
        contentValues.put("date", date);
        contentValues.put("time", time);

        float result = database.insert("tbl_reminder", null,
contentValues);    //returns -1 if data successfully inserts into database

        if (result == -1) {
            return "Failed";
```

```
        } else {
            return "Successfully inserted";
        }

    }

    public String updatereminder(String title, String newTitle, String
date, String time){
        SQLiteDatabase database = this.getWritableDatabase();

        ContentValues contentValues = new ContentValues();
        contentValues.put("title", newTitle);//Updates data into sqlite
database
        contentValues.put("date", date);
        contentValues.put("time", time);

        float result = database.update("tbl_reminder", contentValues,
"title=?", new String[]{title});    //returns -1 if data successfully
inserts into database

        if (result == -1) {
            return "Failed";
        } else {
            return "Successfully updated";
        }
    }

    public String deletereminder(String title){
        SQLiteDatabase database = this.getWritableDatabase();

        float result = database.delete("tbl_reminder", "title=?", new
String[]{title});  //returns -1 if data successfully inserts into database

        if (result == -1) {
            return "Failed";
        } else {
            return "Successfully deleted";
        }
    }



    public Cursor readallreminders() {
        SQLiteDatabase database = this.getWritableDatabase();
        String query = "select * from tbl_reminder order by id desc";
//Sql query to  retrieve  data from the database
        Cursor cursor = database.rawQuery(query, null);
        return cursor;
    }
}
```

Model.java

```
package com.example.carbook.reminders;

//model class is used to set and get the data from the database
public class Model {
    String  title, date, time;
    public Model() {
    }
```

```
    public Model(String title, String date, String time) {
        this.title = title;
        this.date = date;
        this.time = time;
    }
    public String getTitle() {
        return title;
    }
    public void setTitle(String title) {
        this.title = title;
    }
    public String getDate() {
        return date;
    }
    public void setDate(String date) {
        this.date = date;
    }
    public String getTime() {
        return time;
    }
    public void setTime(String time) {
        this.time = time;
    }

}
```

myAdapter.java

```
package com.example.carbook.reminders;

import android.annotation.SuppressLint;
import android.app.Activity;
import android.content.Context;
import android.content.Intent;
import android.graphics.Paint;
import android.view.LayoutInflater;
import android.view.MenuItem;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;

import androidx.annotation.NonNull;
import androidx.appcompat.widget.PopupMenu;
import androidx.recyclerview.widget.RecyclerView;

import com.example.carbook.R;

import java.util.ArrayList;

public class myAdapter extends RecyclerView.Adapter<myAdapter.myviewholder>
{
    ArrayList<Model> dataholder = new ArrayList<Model>();
    Context context;
    Activity activity;
    //array list to hold the reminders

    public myAdapter(Activity activity, Context context, ArrayList<Model>
dataholder) {
        this.dataholder = dataholder;
        this.context=context;
```

```
        this.activity=activity;
    }


    @NonNull
    @Override
    public myviewholder onCreateViewHolder(@NonNull ViewGroup parent, int
viewType) {
        View view =
LayoutInflater.from(parent.getContext()).inflate(R.layout.single_reminder_f
ile, parent, false);
        LayoutInflater layoutInflater = LayoutInflater.from(context);


        //inflates the xml file in recyclerview
        return new myviewholder(view);
    }

    @Override
    public void onBindViewHolder(@NonNull myviewholder holder,
@SuppressLint("RecyclerView") int position) {

        //Binds the single reminder objects to recycler view
        holder.mTitle.setText(dataholder.get(position).getTitle());
        holder.mDate.setText(dataholder.get(position).getDate());
        holder.mTime.setText(dataholder.get(position).getTime());


        holder.itemView.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Intent intent = new Intent(context,
UpdateRemindersActivity.class);
                intent.putExtra("title",
String.valueOf(dataholder.get(position).getTitle()));
                //Starts the new activity to edit reminder
                //activity.startActivityForResult(intent,1);
                activity.startActivity(intent);
            }
        });

        holder.itemView.setOnLongClickListener(new
View.OnLongClickListener() {
            @Override
            public boolean onLongClick(View view) {

                PopupMenu menu = new PopupMenu(view.getContext(), view);
                menu.getMenu().add(R.string.rem_compl_menu);
                menu.setOnMenuItemClickListener(new
PopupMenu.OnMenuItemClickListener() {
                    @Override
                    public boolean onMenuItemClick(MenuItem item) {
                        if(item.getTitle().equals("Completed")||
item.getTitle().equals("Ολοκληρώθηκε"))
                        {
                            //Completed item
                            TextView textTitle,textDate, textTime;
                            textTitle =
holder.itemView.findViewById(R.id.txtTitle);
                            textDate =
holder.itemView.findViewById(R.id.txtDate);
```

```
                                textTime =
holder.itemView.findViewById(R.id.txtTime);
                                if(!textTitle.getPaint().isStrikeThruText()) {

textTitle.setPaintFlags(textTitle.getPaintFlags() |
Paint.STRIKE_THRU_TEXT_FLAG);

textDate.setPaintFlags(textDate.getPaintFlags() |
Paint.STRIKE_THRU_TEXT_FLAG);

textTime.setPaintFlags(textTime.getPaintFlags() |
Paint.STRIKE_THRU_TEXT_FLAG);
                                } else {

textTitle.setPaintFlags(textTitle.getPaintFlags() &
~Paint.STRIKE_THRU_TEXT_FLAG);

textDate.setPaintFlags(textDate.getPaintFlags() &
~Paint.STRIKE_THRU_TEXT_FLAG);

textTime.setPaintFlags(textTime.getPaintFlags() &
~Paint.STRIKE_THRU_TEXT_FLAG);
                                }

                            }

                            return true;
                        }

                    });
                    menu.show();
                    return true;
                }
            });
        }

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

        class myviewholder extends RecyclerView.ViewHolder {

            TextView mTitle, mDate, mTime;


            public myviewholder(@NonNull View itemView) {
                super(itemView);

                mTitle = itemView.findViewById(R.id.txtTitle);
//holds the reference of the materials to show data in recyclerview
                mDate = itemView.findViewById(R.id.txtDate);
                mTime = itemView.findViewById(R.id.txtTime);

            }
        }
}
```

NotificationMessage.java

```
package com.example.carbook.reminders;

import android.os.Bundle;
import android.widget.TextView;

import androidx.appcompat.app.ActionBar;
import androidx.appcompat.app.AppCompatActivity;

import com.example.carbook.R;

//this class creates the Reminder Notification Message
public class NotificationMessage extends AppCompatActivity {
    TextView textView;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_notification_message);
        textView = findViewById(R.id.tv_message);
        Bundle bundle = getIntent().getExtras();
        //call the data which is passed by another intent
        textView.setText(bundle.getString("message"));
    }
}
```

ReminderActivity.java

```
package com.example.carbook.reminders;

import androidx.appcompat.app.ActionBar;
import androidx.appcompat.app.AppCompatActivity;

import android.content.SharedPreferences;
import android.os.Bundle;


import android.app.AlarmManager;
import android.app.DatePickerDialog;
import android.app.PendingIntent;
import android.app.TimePickerDialog;
import android.content.Context;
import android.content.Intent;
import android.view.View;
import android.widget.Button;
import android.widget.DatePicker;
import android.widget.EditText;
import android.widget.TimePicker;
import android.widget.Toast;

import com.example.carbook.mainactivity.MainActivity;
import com.example.carbook.R;

import java.text.DateFormat;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Calendar;
import java.util.Date;
```

```
//this class is to add the take the reminders from the user and inserts
into database
public class ReminderActivity extends AppCompatActivity {

    Button mSubmitbtn, mDatebtn, mTimebtn;
    EditText mTitledit;
    String timeTonotify;
    Boolean notification;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_reminder);
        ActionBar titleS = getSupportActionBar();
        titleS.hide();


        mTitledit = (EditText) findViewById(R.id.editTitle);
        mDatebtn = (Button) findViewById(R.id.btnDate);
        //assigned all the material reference to get and set data
        mTimebtn = (Button) findViewById(R.id.btnTime);
        mSubmitbtn = (Button) findViewById(R.id.btnSbumit);


        mTimebtn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                selectTime();
                //when we click on the choose time button it calls the
select time method
            }
        });

        mDatebtn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                selectDate();
            }
            //when we click on the choose date button it calls the select
date method
        });

        mSubmitbtn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                String title = mTitledit.getText().toString().trim();
                //access the data form the input field
                String date = mDatebtn.getText().toString().trim();
                //access the date form the choose date button
                String time = mTimebtn.getText().toString().trim();
                //access the time form the choose time button

                if (title.isEmpty()) {
                    Toast.makeText(getApplicationContext(),
getString(R.string.empty_field_remind), Toast.LENGTH_SHORT).show();
                    //shows the toast if input field is empty
                } else {
                    if (time.equals("time") || date.equals("date") ||
time.equals("Ωρα") || date.equals("Ημερομηνία")) {
                        //shows toast if date and time are not selected
                        Toast.makeText(getApplicationContext(),
```

```
getString(R.string.select_dateTime), Toast.LENGTH_SHORT).show();
                    } else {
                        processinsert(title, date, time);

                    }
                }


            }
        });
    }


    private void processinsert(String title, String date, String time) {
        SharedPreferences n = getSharedPreferences("Notification_prf",
Context.MODE_PRIVATE);
        notification = n.getBoolean("notif", true);
        String result = new dbManager(this).addreminder(title, date, time);
        //inserts the title,date,time into sql lite database
        if (notification) {
            setAlarm(title, date, time);
            //calls the set alarm method to set alarm
        }else {
            Toast.makeText(getApplicationContext(),
getString(R.string.preference_msg_disabled_notif),
Toast.LENGTH_SHORT).show();
            Intent intentBack = new Intent(getApplicationContext(),
MainActivity.class);
            //this intent will be called once the setting alarm completes
            intentBack.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK |
Intent.FLAG_ACTIVITY_CLEAR_TASK);
            startActivity(intentBack);
        }

        mTitledit.setText("");
        //Toast.makeText(getApplicationContext(), result,
Toast.LENGTH_SHORT).show();
    }

    private void selectTime() {
        //this method performs the time picker task
        Calendar calendar = Calendar.getInstance();
        int hour = calendar.get(Calendar.HOUR_OF_DAY);
        int minute = calendar.get(Calendar.MINUTE);
        TimePickerDialog timePickerDialog = new TimePickerDialog(this, new
TimePickerDialog.OnTimeSetListener() {
            @Override
            public void onTimeSet(TimePicker timePicker, int i, int i1) {
                timeTonotify = i + ":" + i1;
                //temp variable to store the time to set alarm
                mTimebtn.setText(FormatTime(i, i1));
                //sets the button text as selected time
            }
        }, hour, minute, false);
        timePickerDialog.setButton(TimePickerDialog.BUTTON_POSITIVE,
getString(R.string.time_picker_ok_btn),timePickerDialog);
        timePickerDialog.setButton(TimePickerDialog.BUTTON_NEGATIVE,
getString(R.string.time_picker_cancel_btn),timePickerDialog);
        timePickerDialog.show();
    }
```

```
    private void selectDate() {
        //this method performs the date picker task
        Calendar calendar = Calendar.getInstance();
        int year = calendar.get(Calendar.YEAR);
        int month = calendar.get(Calendar.MONTH);
        int day = calendar.get(Calendar.DAY_OF_MONTH);
        DatePickerDialog datePickerDialog = new DatePickerDialog(this, new
DatePickerDialog.OnDateSetListener() {
            @Override
            public void onDateSet(DatePicker datePicker, int year, int
month, int day) {
                mDatebtn.setText(day + "-" + (month + 1) + "-" + year);
                //sets the selected date as test for button
            }
        }, year, month, day);
        datePickerDialog.setButton(TimePickerDialog.BUTTON_POSITIVE,
getString(R.string.time_picker_ok_btn),datePickerDialog);
        datePickerDialog.setButton(TimePickerDialog.BUTTON_NEGATIVE,
getString(R.string.time_picker_cancel_btn),datePickerDialog);
        datePickerDialog.show();
    }

    public String FormatTime(int hour, int minute) {
        //this method converts the time into 12hr farmat and assigns am or
pm

        String time;
        time = "";
        String formattedMinute;

        if (minute / 10 == 0) {
            formattedMinute = "0" + minute;
        } else {
            formattedMinute = "" + minute;
        }


        if (hour == 0) {
            time = "12" + ":" + formattedMinute + " AM";
        } else if (hour < 12) {
            time = hour + ":" + formattedMinute + " AM";
        } else if (hour == 12) {
            time = "12" + ":" + formattedMinute + " PM";
        } else {
            int temp = hour - 12;
            time = temp + ":" + formattedMinute + " PM";
        }


        return time;
    }


    private void setAlarm(String text, String date, String time) {
        AlarmManager am = (AlarmManager)
getSystemService(Context.ALARM_SERVICE);
        //assigining alaram manager object to set alaram

        Intent intent = new Intent(getApplicationContext(),
AlarmBroadcast.class);
        intent.putExtra("event", text);
```

```
        //sending data to alarm class to create channel and notification
        intent.putExtra("time", date);
        intent.putExtra("date", time);

        PendingIntent pendingIntent =
PendingIntent.getBroadcast(getApplicationContext(), 0, intent,
PendingIntent.FLAG_ONE_SHOT);
        String dateandtime = date + " " + timeTonotify;
        DateFormat formatter = new SimpleDateFormat("d-M-yyyy hh:mm");
        try {
            Date date1 = formatter.parse(dateandtime);
            am.set(AlarmManager.RTC_WAKEUP, date1.getTime(),
pendingIntent);
            Toast.makeText(getApplicationContext(),
getString(R.string.rem_set_success_msg), Toast.LENGTH_SHORT).show();

        } catch (ParseException e) {
            e.printStackTrace();
        }

        Intent intentBack = new Intent(getApplicationContext(),
MainActivity.class);
        //this intent will be called once the setting alarm completes
        intentBack.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK |
Intent.FLAG_ACTIVITY_CLEAR_TASK);
        startActivity(intentBack);
        //navigates from adding reminder activity ot mainactivity

    }
}
```

ReminderFragment.java

```
package com.example.carbook.reminders;

import android.content.Intent;
import android.database.Cursor;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;

import androidx.annotation.Nullable;
import androidx.appcompat.app.AppCompatActivity;
import androidx.fragment.app.Fragment;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

import com.example.carbook.R;
import com.example.carbook.mycars.dbCarManager;
import
com.google.android.material.floatingactionbutton.FloatingActionButton;

import java.util.ArrayList;

public class RemindersFragment extends Fragment {

    FloatingActionButton mCreateRem;
    RecyclerView mRecyclerview;
```

```
    ArrayList<Model> dataholder = new ArrayList<Model>();
    //Array list to add reminders and display in recyclerview
    myAdapter adapter;

    TextView tvEmpty;

    @Nullable
    @Override
    public View onCreateView(LayoutInflater inflater, @Nullable  ViewGroup
container, @Nullable  Bundle savedInstanceState) {

        View view = inflater.inflate(R.layout.fragment_reminders,
container, false );

((AppCompatActivity)getActivity()).getSupportActionBar().setTitle(getString
(R.string.reminders));


        tvEmpty = view.findViewById(R.id.empty3);

        mRecyclerview = (RecyclerView)
view.findViewById(R.id.recyclerView);
        mRecyclerview.setLayoutManager(new
LinearLayoutManager(getContext()));
        mCreateRem = (FloatingActionButton)
view.findViewById(R.id.create_reminder);

        Cursor cursor1 = new dbManager(getContext()).reminderIsEmpty();
        cursor1.moveToFirst();
        int count = cursor1.getInt(0);
        if(count != 0) {
            tvEmpty.setVisibility(View.INVISIBLE);
        } else {tvEmpty.setVisibility(View.VISIBLE);}

        //Floating action button to change activity
        mCreateRem.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Intent intent = new Intent(getContext(),
ReminderActivity.class);
                startActivity(intent);
                //Starts the new activity to add Reminders
            }
        });

        Cursor cursor = new dbManager(getContext()).readallreminders();
//Cursor To Load data From the database
        while (cursor.moveToNext()) {
            Model model = new Model(cursor.getString(1),
cursor.getString(2), cursor.getString(3));
            dataholder.add(model);
        }

        adapter = new myAdapter(getActivity(),getContext(),dataholder);
        mRecyclerview.setAdapter(adapter); //Binds the adapter with
recyclerview
        mRecyclerview.setLayoutManager(new
LinearLayoutManager(getActivity()));

        return view;
    }
```

```
    @Override
    public void onActivityResult(int requestCode, int resultCode, @Nullable
@org.jetbrains.annotations.Nullable Intent data) {
        super.onActivityResult(requestCode, resultCode, data);
        if (requestCode == 1){
            //getActivity().recreate();
        }
    }


}
```

UpdateRemindersActivity.java

```
package com.example.carbook.reminders;

import android.app.AlarmManager;
import android.app.AlertDialog;
import android.app.DatePickerDialog;
import android.app.PendingIntent;
import android.app.TimePickerDialog;
import android.content.Context;
import android.content.DialogInterface;
import android.content.Intent;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.DatePicker;
import android.widget.EditText;
import android.widget.TimePicker;
import android.widget.Toast;

import androidx.annotation.Nullable;
import androidx.appcompat.app.ActionBar;
import androidx.appcompat.app.AppCompatActivity;

import com.example.carbook.mainactivity.MainActivity;
import com.example.carbook.R;

import java.text.DateFormat;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Calendar;
import java.util.Date;


public class UpdateRemindersActivity extends AppCompatActivity {

    EditText title_input;
    Button update_button, delete_button, mDatebtn, mTimebtn;
    String title, time1, date1,t,d;
    String timeTonotify;
    Boolean notification;

    @Override
    protected void onCreate(@Nullable @org.jetbrains.annotations.Nullable
Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
```

```
        setContentView(R.layout.activity_update_reminder);
        ActionBar titleS = getSupportActionBar();
        titleS.hide();

        title_input = findViewById(R.id.editTitle1);
        mTimebtn = findViewById(R.id.btnTime1);
        mDatebtn = findViewById(R.id.btnDate1);
        update_button = findViewById(R.id.btnSbumit1);
        delete_button = findViewById(R.id.btndeleterem);

        mTimebtn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                t=selectTime();
            }
        });

        mDatebtn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                d=selectDate();
            }
        });

        //First we call this to get the data
        getSetIntentData();

        update_button.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {

                String newTitle = title_input.getText().toString().trim();
//New setted title
                //access the data form the input field
                String date = mDatebtn.getText().toString().trim();
                //access the date form the choose date button
                String time = mTimebtn.getText().toString().trim();
                //access the time form the choose time button

                if (newTitle.isEmpty()) {
                    Toast.makeText(getApplicationContext(),
getString(R.string.empty_field_remind), Toast.LENGTH_SHORT).show();
                    //shows the toast if input field is empty
                } else {
                    if (time.equals("time") || date.equals("date") ||
time.equals("Ωρα") || date.equals("Ημερομηνία")) {
                        //shows toast if date and time are not selected
                        Toast.makeText(getApplicationContext(),
getString(R.string.select_dateTime), Toast.LENGTH_SHORT).show();
                    } else {
                        SharedPreferences n =
getSharedPreferences("Notification_prf", Context.MODE_PRIVATE);
                        notification = n.getBoolean("notif", true);
                        String result = new
dbManager(UpdateRemindersActivity.this).updatereminder(title,newTitle,date,
time);
                        if (notification) {
                        setAlarm(newTitle, date, time);
                        }else {
                            Toast.makeText(getApplicationContext(),
getString(R.string.preference_msg_disabled_notif),
```

```
Toast.LENGTH_SHORT).show();
                            Intent intentBack = new
Intent(getApplicationContext(), MainActivity.class);
                            //this intent will be called once the setting
alarm completes

intentBack.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK |
Intent.FLAG_ACTIVITY_CLEAR_TASK);
                            startActivity(intentBack);
                        }


                    }
                }

            }
        });

        delete_button.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                confirmDialog();
                //Να δω αν μπορώ να γυρίσω στο fragment
            }
        });




    }


    void getSetIntentData(){
        if(getIntent().hasExtra("title")){
            //Getting data from intent
            title = getIntent().getStringExtra("title");  //Old setted
title

            //Setting intent data
            title_input.setText(title);
        }else{
            Toast.makeText(this, getString(R.string.rem_update_error),
Toast.LENGTH_SHORT).show();
        }
        //return title;

    }

    void confirmDialog(){
        AlertDialog.Builder builder = new AlertDialog.Builder(this);
        builder.setMessage(R.string.delete_rem);
        builder.setPositiveButton(R.string.btn_delete_remdialog, new
DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialogInterface, int i) {
                String result = new
dbManager(UpdateRemindersActivity.this).deletereminder(title);
                Toast.makeText(getApplicationContext(),
getString(R.string.rem_del_succes_msg), Toast.LENGTH_SHORT).show();
                finish();
                Intent intentBack = new Intent(getApplicationContext(),
```

```
MainActivity.class);
                //this intent will be called once the setting alarm
completes
                intentBack.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK |
Intent.FLAG_ACTIVITY_CLEAR_TASK);
                startActivity(intentBack);
            }
        });
        builder.setNegativeButton(R.string.btn_cancel_remdialog, new
DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialogInterface, int i) {
                Toast.makeText(getApplicationContext(),
getString(R.string.del_cancel_remdialog), Toast.LENGTH_SHORT).show();
            }
        });
        builder.create().show();
    }

    private String selectTime() {
        //this method performs the time picker task
        Calendar calendar = Calendar.getInstance();
        int hour = calendar.get(Calendar.HOUR_OF_DAY);
        int minute = calendar.get(Calendar.MINUTE);
        TimePickerDialog timePickerDialog = new TimePickerDialog(this, new
TimePickerDialog.OnTimeSetListener() {
            @Override
            public void onTimeSet(TimePicker timePicker, int i, int i1) {
                timeTonotify = i + ":" + i1;
                //temp variable to store the time to set alarm
                mTimebtn.setText(FormatTime(i, i1));
                date1= (String) mTimebtn.getText();
                //sets the button text as selected time
            }
        }, hour, minute, false);
        timePickerDialog.setButton(TimePickerDialog.BUTTON_POSITIVE,
getString(R.string.time_picker_ok_btn),timePickerDialog);
        timePickerDialog.setButton(TimePickerDialog.BUTTON_NEGATIVE,
getString(R.string.time_picker_cancel_btn),timePickerDialog);
        timePickerDialog.show();
        return date1;
    }

    private String selectDate() {
        //this method performs the date picker task
        Calendar calendar = Calendar.getInstance();
        int year = calendar.get(Calendar.YEAR);
        int month = calendar.get(Calendar.MONTH);
        int day = calendar.get(Calendar.DAY_OF_MONTH);
        DatePickerDialog datePickerDialog = new DatePickerDialog(this, new
DatePickerDialog.OnDateSetListener() {
            @Override
            public void onDateSet(DatePicker datePicker, int year, int
month, int day) {
                mDatebtn.setText(day + "-" + (month + 1) + "-" + year);
                time1= (String) mDatebtn.getText();
                //sets the selected date as test for button
            }
        }, year, month, day);
        datePickerDialog.setButton(TimePickerDialog.BUTTON_POSITIVE,
getString(R.string.time_picker_ok_btn),datePickerDialog);
```

```
        datePickerDialog.setButton(TimePickerDialog.BUTTON_NEGATIVE,
getString(R.string.time_picker_cancel_btn),datePickerDialog);
        datePickerDialog.show();
        return time1;
    }

    public String FormatTime(int hour, int minute) {
        //this method converts the time into 12hr farmat and assigns am or
pm

        String time;
        String formattedMinute;

        if (minute / 10 == 0) {
            formattedMinute = "0" + minute;
        } else {
            formattedMinute = "" + minute;
        }


        if (hour == 0) {
            time = "12" + ":" + formattedMinute + " AM";
        } else if (hour < 12) {
            time = hour + ":" + formattedMinute + " AM";
        } else if (hour == 12) {
            time = "12" + ":" + formattedMinute + " PM";
        } else {
            int temp = hour - 12;
            time = temp + ":" + formattedMinute + " PM";
        }


        return time;
    }

    private void setAlarm(String text, String date, String time) {
        AlarmManager am = (AlarmManager)
getSystemService(Context.ALARM_SERVICE);
        //assigining alaram manager object to set alaram

        Intent intent = new Intent(getApplicationContext(),
AlarmBroadcast.class);
        intent.putExtra("event", text);
        //sending data to alarm class to create channel and notification
        intent.putExtra("time", date);
        intent.putExtra("date", time);

        PendingIntent pendingIntent =
PendingIntent.getBroadcast(getApplicationContext(), 0, intent,
PendingIntent.FLAG_ONE_SHOT);
        String dateandtime = date + " " + timeTonotify;
        DateFormat formatter = new SimpleDateFormat("d-M-yyyy hh:mm");
        try {
            Date date1 = formatter.parse(dateandtime);
            am.set(AlarmManager.RTC_WAKEUP, date1.getTime(),
pendingIntent);
            Toast.makeText(getApplicationContext(),
getString(R.string.rem_set_success_msg), Toast.LENGTH_SHORT).show();

        } catch (ParseException e) {
            e.printStackTrace();
```

```
        }

        Intent intentBack = new Intent(getApplicationContext(),
MainActivity.class);
        //this intent will be called once the setting alarm completes
        intentBack.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK |
Intent.FLAG_ACTIVITY_CLEAR_TASK);
        startActivity(intentBack);
        //navigates from adding reminder activity ot mainactivity
    }
}
```

CarServiceModel.java

```
package com.example.carbook.servicebook;

public class CarServiceModel {
    int visit_code;
    Long task_code;

    public CarServiceModel(int visit_code, Long task_code) {
        this.visit_code = visit_code;
        this.task_code = task_code;
    }

    public int getVisit_code() {
        return visit_code;
    }

    public void setVisit_code(int visit_code) {
        this.visit_code = visit_code;
    }

    public Long getTask_code() {
        return task_code;
    }

    public void setTask_code(Long task_code) {
        this.task_code = task_code;
    }
}
```

CarVisitActivity.java

```
package com.example.carbook.servicebook;

import android.app.AlarmManager;
import android.app.PendingIntent;
import android.content.Context;
import android.content.DialogInterface;
import android.content.Intent;
import android.content.SharedPreferences;
import android.database.Cursor;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;
```

```
import androidx.appcompat.app.AlertDialog;
import androidx.appcompat.app.AppCompatActivity;

import com.android.volley.Request;
import com.android.volley.Response;
import com.android.volley.VolleyError;
import com.android.volley.toolbox.JsonArrayRequest;
import com.android.volley.toolbox.JsonObjectRequest;
import com.example.carbook.R;
import com.example.carbook.mainactivity.MainActivity;
import com.example.carbook.mainactivity.RequestSingleton;
import com.example.carbook.mycars.VolleyErrorHelper;
import com.example.carbook.mycars.dbCarManager;
import com.example.carbook.myworkshops.WorkshopsModel;
import com.example.carbook.reminders.AlarmBroadcast;
import com.example.carbook.reminders.dbManager;
import
com.google.android.material.floatingactionbutton.FloatingActionButton;

import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

import java.text.DateFormat;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;

public class CarVisitActivity extends AppCompatActivity {

    long speedometer;
    long odometer;
    long nextService;
    long w_code;
    Long modelcode, myCarId;
    int min_month,p_day, p_month,p_year;
    String pDate, pDate1,pDateNotYet, Model,Brand, Chassis,d,m,y,o;
    Boolean notification, bool,emCheck=false;
    int wsPosition = 0;

    ArrayList<WorkshopsModel> dataholder = new ArrayList<WorkshopsModel>();

    TextView tvTasks,tvWorkshops,tvNextService,tvNextServiceMonths,
tvHiddenTasks,tvHiddenTasks2,tvHiddenWS, tvHiddenWSId,tvSelectedTasksBool;
    EditText etOdometer, etDateDay,etDateMonth, etDateYear;
    FloatingActionButton myVisitSave;
    Button showNextBtn;

    boolean[] selectedTasks;
    ArrayList<Integer> taskList = new ArrayList<>();
    ArrayList<String> taskArray1 = new ArrayList<>();
    String[] wsArray = new String[9];
    ArrayList<VisitScheduleModel> visitScheduleList = new ArrayList<>();


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_car_visit);
```

```
        tvTasks = findViewById(R.id.s_t_by_vs);
        tvWorkshops = findViewById(R.id.sb_workshops_by_tasks);
        tvNextService = findViewById(R.id.sb_next_srv);
        tvNextServiceMonths = findViewById(R.id.sb_next_srv_months);
        tvHiddenTasks = findViewById(R.id.tasks_hidden);
        tvHiddenTasks2 = findViewById(R.id.tv_hidden_tasks2);
        tvHiddenWS = findViewById(R.id.tv_hidden_ws);
        tvHiddenWSId = findViewById(R.id.ws_id_hidden);
        tvSelectedTasksBool = findViewById(R.id.seleted_tasks_bool);

        etOdometer = findViewById(R.id.sb_speedometer);
        etDateDay = findViewById(R.id.visit_date_day);
        etDateMonth = findViewById(R.id.visit_date_month);
        etDateYear = findViewById(R.id.visit_date_year);

        myVisitSave = findViewById(R.id.my_visit_save);
        showNextBtn = findViewById(R.id.next_visit_btn);

        bool=false;

        getSetIntentData();




        showNextBtn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {

                d = etDateDay.getText().toString().trim();
                m = etDateMonth.getText().toString().trim();
                y = etDateYear.getText().toString().trim();
                o = etOdometer.getText().toString().trim();

                visitScheduleList.clear();
                tvHiddenTasks.setText("");
                emCheck=false;


                tvHiddenTasks2.setText("");

                if ((etOdometer.getText().toString().trim().isEmpty()) ||
(etDateDay.getText().toString().trim().isEmpty())
                        ||
(etDateMonth.getText().toString().trim().isEmpty()) ||
(etDateYear.getText().toString().trim().isEmpty())
                        ||
(tvWorkshops.getText().toString().trim().isEmpty()) ||
(tvTasks.getText().toString().trim().isEmpty())) {

                    Toast.makeText(CarVisitActivity.this,
getString(R.string.car_all_fields), Toast.LENGTH_SHORT).show();
                } else if ((!d.matches("\\d+(?:\\.\\d+)?")) ||
(!m.matches("\\d+(?:\\.\\d+)?")) || (!y.matches("\\d+(?:\\.\\d+)?")) ||
(!o.matches("\\d+(?:\\.\\d+)?"))) {

                    Toast.makeText(getApplicationContext(),
getString(R.string.Wrong_data_input_msg_mycars),
Toast.LENGTH_SHORT).show();
```

```
                } else {

                odometer =
Long.parseLong(etOdometer.getText().toString().trim());
                p_day =
Integer.parseInt(etDateDay.getText().toString().trim());
                p_month =
Integer.parseInt(etDateMonth.getText().toString().trim());
                p_year =
Integer.parseInt(etDateYear.getText().toString().trim());

                    if(odometer==0L || odometer<5000) {
                        String url =
"http://192.168.1.3:8080/api/bymodel?modelCode=" + modelcode; //For android
phone

                        JsonArrayRequest jsonArrayRequest = new
JsonArrayRequest(Request.Method.GET, url, null, new
Response.Listener<JSONArray>() {
                            @Override
                            public void onResponse(JSONArray response) {

                                try {
                                    JSONArray jsonArray = new
JSONArray(response.toString());
                                    Long min = 3000000L;
                                    //min_month = 1000;
                                    for (int i = 0; i < jsonArray.length();
i++) {
                                        JSONObject jsonObject =
jsonArray.getJSONObject(i);
                                        Long taskCode =
jsonObject.optLong("taskCode");
                                        Long kilometers =
jsonObject.optLong("kilometers");
                                        int months =
jsonObject.optInt("months");
                                        if (kilometers < min) {
                                            min = kilometers;
                                            //min_month = months;
                                        }
                                        JSONObject jsonObject1 =
jsonObject.getJSONObject("vs");
                                        Long sCode =
jsonObject1.getLong("sCode");
                                        Long modelCode2 =
jsonObject1.getLong("modelCode");
                                        visitScheduleList.add(new
VisitScheduleModel(sCode, modelCode2, taskCode, kilometers, months));
                                    }


                                    for (int i = 0; i <
visitScheduleList.size(); i++) {

                                        int nextTasks = (int) (min /
visitScheduleList.get(i).getKilometers());
                                        nextTasks = Math.round(nextTasks);
```

```
//Rounded towards up if nextTasks==2,9 -> 2 rounded

                                        String tsk=
tvTasks.getText().toString().trim();
                                        String[] tskArray =
tsk.split("\n");
                                        final int[] check = {0};


                                        if (nextTasks == 1) {

                                            String url =
"http://192.168.1.3:8080/api/cartask/" +
visitScheduleList.get(i).getTaskCode();
                                            int finalI = i;
                                            Long finalMin = min;
                                            JsonObjectRequest
jsonObjectRequest = new JsonObjectRequest(Request.Method.GET, url, null,
new Response.Listener<JSONObject>() {
                                                @Override
                                                public void
onResponse(JSONObject response) {

                                                    if
(finalMin.equals(visitScheduleList.get(finalI).getKilometers())) {  //First
service
                                                        String taskPart =
response.optString("taskPart");
                                                        String taskType =
response.optString("taskType");


                                                        for(int
i=0;i<tskArray.length;i++) {
                                                            if
(taskPart.equals(tskArray[i])){
                                                                check[0]
=1;
emCheck=true;
                                                                String
tvText = tvHiddenTasks.getText().toString().trim();
                                                                String
stringBuilder = tvText + " \n" + taskPart + " (" + taskType + ")";
tvHiddenTasks.setText(stringBuilder);
                                                            }
                                                        }

                                                        if(check[0]==0) {

                                                            String tvText =
tvHiddenTasks2.getText().toString().trim();
                                                            String
stringBuilder = tvText + " \n" + taskPart + " (" + taskType + ")";
tvHiddenTasks2.setText(stringBuilder);

                                                        }
```

```
                                                    }

                                                }
                                            }, new Response.ErrorListener()
{
                                                @Override
                                                public void
onErrorResponse(VolleyError error) {
                                                    String errormsg;
                                                    errormsg =
VolleyErrorHelper.getMessage(error, CarVisitActivity.this);

Toast.makeText(CarVisitActivity.this, errormsg, Toast.LENGTH_SHORT).show();
                                                }
                                            });

RequestSingleton.getInstance(CarVisitActivity.this).addToRequestQueue(jsonO
bjectRequest);
                                        }

                                    }

                                } catch (JSONException e) {
                                    e.printStackTrace();
                                }

                            }
                        }, new Response.ErrorListener() {
                            @Override
                            public void onErrorResponse(VolleyError error)
{

                                String errormsg;
                                errormsg =
VolleyErrorHelper.getMessage(error, CarVisitActivity.this);
                                Toast.makeText(CarVisitActivity.this,
errormsg, Toast.LENGTH_SHORT).show();

                            }
                        });


                        //Add the json request object to the request queue

RequestSingleton.getInstance(CarVisitActivity.this).addToRequestQueue(jsonA
rrayRequest);
                    }


                //Request from database
                String url =
"http://192.168.1.3:8080/api/bymodel?modelCode=" + modelcode; //For android
phone

                JsonArrayRequest jsonArrayRequest = new
JsonArrayRequest(Request.Method.GET, url, null, new
Response.Listener<JSONArray>() {
                    @Override
                    public void onResponse(JSONArray response) {

                        try {
```

```
                            JSONArray jsonArray = new
JSONArray(response.toString());
                            Long min = 3000000L;
                            min_month = 1000;
                            for (int i = 0; i < jsonArray.length(); i++) {
                                JSONObject jsonObject =
jsonArray.getJSONObject(i);
                                Long taskCode =
jsonObject.optLong("taskCode");
                                Long kilometers =
jsonObject.optLong("kilometers");
                                int months = jsonObject.optInt("months");
                                if (kilometers < min) {
                                    min = kilometers;
                                    min_month = months;
                                }
                                JSONObject jsonObject1 =
jsonObject.getJSONObject("vs");
                                Long sCode = jsonObject1.getLong("sCode");
                                Long modelCode2 =
jsonObject1.getLong("modelCode");
                                visitScheduleList.add(new
VisitScheduleModel(sCode, modelCode2, taskCode, kilometers, months));
                            }
                            nextService = odometer + min;
                            pDate1 = p_day + "-" + p_month + "-" + p_year;
                            pDateNotYet = p_day+2 + "-" + p_month + "-" +
p_year;

tvNextService.setText(String.valueOf(nextService));
                            if (min_month == 12) {
                                p_year += 1;
                            } else if (min_month == 24) {
                                p_year += 2;
                            } else if (min_month == 36) {
                                p_year += 3;
                            } else if (min_month == 48) {
                                p_year += 4;
                            }
                            pDate = p_day + "-" + p_month + "-" + p_year;
                            tvNextServiceMonths.setText(pDate);
                            p_month -= 1;
                            pDate = p_day + "-" + p_month + "-" + p_year;
                            bool = true;




                        } catch (JSONException e) {
                            e.printStackTrace();
                        }
                        String stringBuilder1 = " \n";
                        tvHiddenTasks.setText(stringBuilder1);
                        tvHiddenTasks2.setText(stringBuilder1);


                        for (int i = 0; i < visitScheduleList.size(); i++)
{
                            int nextTasks = (int) (odometer /
visitScheduleList.get(i).getKilometers());
```

```
                            nextTasks = Math.round(nextTasks); //Rounded
towards up if nextTasks==2,9 -> 2 rounded

                            String tsk=
tvTasks.getText().toString().trim();
                            String[] tskArray = tsk.split("\n");
                            final int[] check = {0};


                            if ((nextTasks == 1 || nextTasks == 0)) {


                                String url =
"http://192.168.1.3:8080/api/cartask/" +
visitScheduleList.get(i).getTaskCode();
                                int finalI = i;
                                JsonObjectRequest jsonObjectRequest = new
JsonObjectRequest(Request.Method.GET, url, null, new
Response.Listener<JSONObject>() {
                                    @Override
                                    public void onResponse(JSONObject
response) {

                                        if (((odometer) ==
visitScheduleList.get(finalI).getKilometers())) {  //First service
                                            String taskPart =
response.optString("taskPart");
                                            String taskType =
response.optString("taskType");


                                            for(int
i=0;i<tskArray.length;i++) {
                                                if
(taskPart.equals(tskArray[i])){
                                                    check[0] =1;
                                                    emCheck=true;
                                                    String tvText =
tvHiddenTasks.getText().toString().trim();
                                                    String stringBuilder =
tvText + " \n" + taskPart + " (" + taskType + ")";
                                                    tvHiddenTasks.setText(stringBuilder);
                                                }
                                            }

                                            if(check[0]==0) {


                                                String tvText =
tvHiddenTasks2.getText().toString().trim();
                                                String stringBuilder =
tvText + " \n" + taskPart + " (" + taskType + ")";
                                                tvHiddenTasks2.setText(stringBuilder);


                                            }


                                        } else if
```

```
((visitScheduleList.get(finalI).getKilometers() - odometer < 3001) &&
odometer - visitScheduleList.get(finalI).getKilometers() > 0) {
                                                String taskPart =
response.optString("taskPart");
                                                String taskType =
response.optString("taskType");

                                                for(int
i=0;i<tskArray.length;i++) {
                                                    if
(taskPart.equals(tskArray[i])){
                                                        check[0] =1;
                                                        emCheck=true;
                                                        String tvText =
tvHiddenTasks.getText().toString().trim();
                                                        String stringBuilder =
tvText + " \n" + taskPart + " (" + taskType + ")";

tvHiddenTasks.setText(stringBuilder);
                                                    }
                                                }

                                                if(check[0]==0) {

                                                    String tvText =
tvHiddenTasks2.getText().toString().trim();
                                                    String stringBuilder =
tvText + " \n" + taskPart + " (" + taskType + ")";

tvHiddenTasks2.setText(stringBuilder);

                                                }

                                            } else if
((visitScheduleList.get(finalI).getKilometers() - odometer < 10001 &&
visitScheduleList.get(finalI).getKilometers() - odometer > -5001) &&
odometer - visitScheduleList.get(finalI).getKilometers() < 0) {
                                                String taskPart =
response.optString("taskPart");
                                                String taskType =
response.optString("taskType");

                                                for(int
i=0;i<tskArray.length;i++) {
                                                    if
(taskPart.equals(tskArray[i])){
                                                        check[0] =1;
                                                        emCheck=true;
                                                        String tvText =
tvHiddenTasks.getText().toString().trim();
                                                        String stringBuilder =
tvText + " \n" + taskPart + " (" + taskType + ")";

tvHiddenTasks.setText(stringBuilder);
                                                    }
                                                }

                                                if(check[0]==0) {

                                                    String tvText =
tvHiddenTasks2.getText().toString().trim();
```

```
                                                String stringBuilder =
tvText + " \n" + taskPart + " (" + taskType + ")";

tvHiddenTasks2.setText(stringBuilder);

                                            }


                                        }
                                    }
                                }, new Response.ErrorListener() {
                                    @Override
                                    public void onErrorResponse(VolleyError
error) {
                                        String errormsg;
                                        errormsg =
VolleyErrorHelper.getMessage(error, CarVisitActivity.this);

Toast.makeText(CarVisitActivity.this, errormsg, Toast.LENGTH_SHORT).show();
                                    }
                                });

RequestSingleton.getInstance(CarVisitActivity.this).addToRequestQueue(jsonO
bjectRequest);


                            } else if (nextTasks == 2) {

                                String url =
"http://192.168.1.3:8080/api/cartask/" +
visitScheduleList.get(i).getTaskCode();
                                JsonObjectRequest jsonObjectRequest = new
JsonObjectRequest(Request.Method.GET, url, null, new
Response.Listener<JSONObject>() {
                                    @Override
                                    public void onResponse(JSONObject
response) {
                                        String taskPart =
response.optString("taskPart");
                                        String taskType =
response.optString("taskType");

                                        for(int i=0;i<tskArray.length;i++)
{
                                            if
(taskPart.equals(tskArray[i])){
                                                check[0] =1;
                                                emCheck=true;
                                                String tvText =
tvHiddenTasks.getText().toString().trim();
                                                String stringBuilder =
tvText + " \n" + taskPart + " (" + taskType + ")";

tvHiddenTasks.setText(stringBuilder);
                                            }
                                        }

                                        if(check[0]==0) {

                                            String tvText =
tvHiddenTasks2.getText().toString().trim();
```

```
                                            String stringBuilder = tvText +
" \n" + taskPart + " (" + taskType + ")";

tvHiddenTasks2.setText(stringBuilder);

                                        }


                                    }
                                }, new Response.ErrorListener() {
                                    @Override
                                    public void onErrorResponse(VolleyError
error) {
                                        String errormsg;
                                        errormsg =
VolleyErrorHelper.getMessage(error, CarVisitActivity.this);

Toast.makeText(CarVisitActivity.this, errormsg, Toast.LENGTH_SHORT).show();
                                    }
                                });

RequestSingleton.getInstance(CarVisitActivity.this).addToRequestQueue(jsonO
bjectRequest);


                            } else if (nextTasks == 4) {


                                String url =
"http://192.168.1.3:8080/api/cartask/" +
visitScheduleList.get(i).getTaskCode();
                                JsonObjectRequest jsonObjectRequest = new
JsonObjectRequest(Request.Method.GET, url, null, new
Response.Listener<JSONObject>() {
                                    @Override
                                    public void onResponse(JSONObject
response) {
                                        String taskPart =
response.optString("taskPart");
                                        String taskType =
response.optString("taskType");


                                        for(int i=0;i<tskArray.length;i++)
{
                                            if
(taskPart.equals(tskArray[i])){
                                                check[0] =1;
                                                emCheck=true;
                                                String tvText =
tvHiddenTasks.getText().toString().trim();
                                                String stringBuilder =
tvText + " \n" + taskPart + " (" + taskType + ")";

tvHiddenTasks.setText(stringBuilder);
                                            }
                                        }

                                        if(check[0]==0) {

                                            String tvText =
```

```
tvHiddenTasks2.getText().toString().trim();
                                        String stringBuilder = tvText +
" \n" + taskPart + " (" + taskType + ")";

tvHiddenTasks2.setText(stringBuilder);

                                    }


                                }
                            }, new Response.ErrorListener() {
                                @Override
                                public void onErrorResponse(VolleyError
error) {
                                    String errormsg;
                                    errormsg =
VolleyErrorHelper.getMessage(error, CarVisitActivity.this);

Toast.makeText(CarVisitActivity.this, errormsg, Toast.LENGTH_SHORT).show();
                                }
                            });

RequestSingleton.getInstance(CarVisitActivity.this).addToRequestQueue(jsonO
bjectRequest);

                        } else if (nextTasks == 6) {


                            String url =
"http://192.168.1.3:8080/api/cartask/" +
visitScheduleList.get(i).getTaskCode();
                            JsonObjectRequest jsonObjectRequest = new
JsonObjectRequest(Request.Method.GET, url, null, new
Response.Listener<JSONObject>() {
                                @Override
                                public void onResponse(JSONObject
response) {
                                    String taskPart =
response.optString("taskPart");
                                    String taskType =
response.optString("taskType");

                                    for(int i=0;i<tskArray.length;i++)
{
                                        if
(taskPart.equals(tskArray[i])){
                                            check[0] =1;
                                            emCheck=true;
                                            String tvText =
tvHiddenTasks.getText().toString().trim();
                                            String stringBuilder =
tvText + " \n" + taskPart + " (" + taskType + ")";

tvHiddenTasks.setText(stringBuilder);
                                        }
                                    }

                                    if(check[0]==0) {

                                        String tvText =
tvHiddenTasks2.getText().toString().trim();
```

```
                                                String stringBuilder = tvText +
" \n" + taskPart + " (" + taskType + ")";

tvHiddenTasks2.setText(stringBuilder);

                                            }


                                        }
                                    }, new Response.ErrorListener() {
                                        @Override
                                        public void onErrorResponse(VolleyError
error) {
                                            String errormsg;
                                            errormsg =
VolleyErrorHelper.getMessage(error, CarVisitActivity.this);

Toast.makeText(CarVisitActivity.this, errormsg, Toast.LENGTH_SHORT).show();
                                        }
                                    });

RequestSingleton.getInstance(CarVisitActivity.this).addToRequestQueue(jsonO
bjectRequest);

                                }
                            }
                            String tvText =
tvHiddenTasks.getText().toString().trim();
                            String stringBuilder = tvText + "\n:";
                            tvHiddenTasks.setText(stringBuilder);

                            String tvText2 =
tvHiddenTasks2.getText().toString().trim();
                            String stringBuilder2 = tvText2 + "\n:";
                            tvHiddenTasks2.setText(stringBuilder2);


                        }
                    }, new Response.ErrorListener() {
                        @Override
                        public void onErrorResponse(VolleyError error) {

                            String errormsg;
                            errormsg = VolleyErrorHelper.getMessage(error,
CarVisitActivity.this);
                            Toast.makeText(CarVisitActivity.this, errormsg,
Toast.LENGTH_SHORT).show();

                        }
                    });


                    //Add the json request object to the request queue

RequestSingleton.getInstance(CarVisitActivity.this).addToRequestQueue(jsonA
rrayRequest);
                    tvHiddenWS.setText("");
                    for (int codeW = 1; codeW < 8; codeW++) {
                        String url1 = "http://192.168.1.3:8080/api/workshop/" +
codeW;
                        JsonObjectRequest jsonObjectRequest1 = new
```

```
JsonObjectRequest(Request.Method.GET, url1, null, new
Response.Listener<JSONObject>() {
                        @Override
                        public void onResponse(JSONObject response) {

                            String name = response.optString("name");
                            String tvText =
tvHiddenWS.getText().toString().trim();
                            String stringBuilder = tvText + " \n" + name;
                            tvHiddenWS.setText(stringBuilder);


                        }
                    }, new Response.ErrorListener() {
                        @Override
                        public void onErrorResponse(VolleyError error) {
                            String errormsg;
                            errormsg = VolleyErrorHelper.getMessage(error,
CarVisitActivity.this);
                            Toast.makeText(CarVisitActivity.this, errormsg,
Toast.LENGTH_SHORT).show();
                        }
                    });

RequestSingleton.getInstance(CarVisitActivity.this).addToRequestQueue(jsonO
bjectRequest1);
                }


                            Toast.makeText(CarVisitActivity.this,
getString(R.string.ypu_will_be_notified), Toast.LENGTH_SHORT).show();

                    }

            }
        });


        myVisitSave.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {


                if (bool) {
                    odometer =
Long.parseLong(etOdometer.getText().toString().trim());  //If the same as
previous not saving tasks
                    nextService =
Long.parseLong(tvNextService.getText().toString().trim());
                    pDate =
tvNextServiceMonths.getText().toString().trim();
                    w_code =
Long.parseLong(tvHiddenWSId.getText().toString().trim());  //Check if ws
selected
                    String tasks = tvTasks.getText().toString().trim();
                    String [] separated = tasks.split("\n");
                    String result2 = new
dbCarManager(CarVisitActivity.this).addVisit(myCarId, odometer,
nextService, w_code, pDate1);
                    Cursor res = new
dbCarManager(CarVisitActivity.this).getVisitCode(odometer);
```

```
                    res.moveToFirst();
                    long visit_code = res.getLong(0);  //Correct value



                    for(int i=0;i<separated.length;i++){  //Initial make
                        String res1 = new
dbCarManager(CarVisitActivity.this).addService(visit_code,separated[i]);
                    }
                    Toast.makeText(getApplicationContext(),
getString(R.string.msg_visit_saved), Toast.LENGTH_SHORT).show();

                    SharedPreferences n =
getSharedPreferences("Notification_prf", Context.MODE_PRIVATE);
                    notification = n.getBoolean("notif", true);
                    String stringTasks =
tvHiddenTasks.getText().toString().trim();
                    String stringWS =
tvHiddenWS.getText().toString().trim();
                    String title = getString(R.string.cv_rem_text) + " " +
Brand + " " + Model + " " + getString(R.string.with_chassis_number) + " " +
Chassis + " " + getString(R.string.cv_rem_part2)+" "+stringTasks +" \n\n"+
getString(R.string.ws_text_rem)+"\n"+ stringWS;
                    String time = "10:00AM";

                    String stringTasks2 =
tvHiddenTasks2.getText().toString().trim();
                    String title2 = getString(R.string.cv_rem_text) + " " +
Brand + " " + Model + " " + getString(R.string.with_chassis_number) + " " +
Chassis + " " + getString(R.string.cv_rem_part2)+" "+stringTasks2 +" \n\n"+
getString(R.string.ws_text_rem)+"\n"+ stringWS;
                    if(emCheck) {
                        String result = new
dbManager(CarVisitActivity.this).addreminder(title, pDate, time);
                    }

                    int vis = (int) visit_code;
                    if (((vis & 1) == 1) && emCheck) {
                        String result1 = new
dbManager(CarVisitActivity.this).addreminder(title2, pDateNotYet, time);
                    }
                    //inserts the title,date,time into sql lite database
                    if (notification) {
                        if(emCheck) {
                            setAlarm(title, pDate, time);
                        }
                        // Condition Check
                        // Bitwise AND of any odd number by 1 gives 1
                        int vi = (int) visit_code;
                        if (((vi & 1) == 1) && emCheck) {
                            setAlarm(title2, pDateNotYet, time);
                        }
                        //calls the set alarm method to set alarm
                    } else {
                        Toast.makeText(getApplicationContext(),
getString(R.string.preference_msg_disabled_notif),
Toast.LENGTH_SHORT).show();
                    }
                    emCheck=false;

                    Intent intentBack = new Intent(getApplicationContext(),
```

```
MainActivity.class);
                    //this intent will be called once the setting alarm
completes
                    intentBack.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK |
Intent.FLAG_ACTIVITY_CLEAR_TASK);
                    startActivity(intentBack);
                    //w_code; Get it somehow
                    //model code from intent
                } else {
                    Toast.makeText(getApplicationContext(),
getString(R.string.click_show_next_first), Toast.LENGTH_SHORT).show();
                }

            }
        });

        tvWorkshops.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {


                //Get Workshops
                String url = "http://192.168.1.3:8080/api/workshops";
                JsonArrayRequest jsonArrayRequest = new
JsonArrayRequest(Request.Method.GET, url, null, new
Response.Listener<JSONArray>() {
                    @Override
                    public void onResponse(JSONArray response) {

                        try {
                            JSONArray jsonArray = new
JSONArray(response.toString());
                            for(int i=0; i<jsonArray.length(); i++){
                                JSONObject jsonObject =
jsonArray.getJSONObject(i);
                                Long code = jsonObject.optLong("id");
                                String name = jsonObject.optString("name");
                                String area = jsonObject.optString("area");
                                String address =
jsonObject.optString("address");
                                String coordinates =
jsonObject.optString("coordinates");
                                String telephone =
jsonObject.optString("telephone");

                                dataholder.add(new
WorkshopsModel(code,name,area,address,coordinates,telephone));
                                wsArray[i] = name;



                            }
                            //Initialize alert dialog
                            AlertDialog.Builder builder = new
AlertDialog.Builder(CarVisitActivity.this);

                            //Set title

builder.setTitle(getString(R.string.choose_workshops));
                            builder.setCancelable(false);
                            builder.setSingleChoiceItems(wsArray,
```

```
wsPosition, new DialogInterface.OnClickListener() {
                                    @Override
                                    public void onClick(DialogInterface dialog,
int which) {
                                        wsPosition = which;

tvHiddenWSId.setText(Integer.toString(which + 1));
                                    }
                                });


builder.setPositiveButton(getString(R.string.dialog_taskList_ok), new
DialogInterface.OnClickListener() {
                                    @Override
                                    public void onClick(DialogInterface dialog,
int which) {

if(!tvHiddenWSId.getText().toString().trim().equals("")) {
                                            //Set text on text view

tvWorkshops.setText(wsArray[wsPosition]);

if(!tvTasks.equals(getString(R.string.select_the_tasks_which_were_perfomed)
)){
                                                tvTasks.setText("");
                                            }
                                        }
                                    }
                                });

builder.setNegativeButton(R.string.sb_cancel_tasks, new
DialogInterface.OnClickListener() {
                                    @Override
                                    public void onClick(DialogInterface dialog,
int which) {
                                        //dialogInterface.dismiss();
                                    }
                                });
                                builder.show();

                            } catch (JSONException e) {
                                e.printStackTrace();
                            }

                        }
                    }, new Response.ErrorListener() {
                        @Override
                        public void onErrorResponse(VolleyError error) {

                            String errormsg;
                            errormsg=
VolleyErrorHelper.getMessage(error,CarVisitActivity.this);
                            Toast.makeText(CarVisitActivity.this, errormsg,
Toast.LENGTH_SHORT).show();

                        }
                    });

                    //Add the json request object to the request queue

RequestSingleton.getInstance(CarVisitActivity.this).addToRequestQueue(jsonA
```

```
rrayRequest);
            }
        });

        tvTasks.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                int wsCode=0;

                if (tvHiddenWSId.getText().toString().trim().equals("")) {
                    Toast.makeText(CarVisitActivity.this,
getString(R.string.choose_ws_first), Toast.LENGTH_SHORT).show();

                } else {
                    if (wsCode !=
Integer.parseInt(tvHiddenWSId.getText().toString().trim())) {

                        taskArray1.clear();

                        String url =
"http://192.168.1.3:8080/api/wstype?codeW=" +
Integer.parseInt(tvHiddenWSId.getText().toString().trim());
                        JsonArrayRequest jsonArrayRequest = new
JsonArrayRequest(Request.Method.GET, url, null, new
Response.Listener<JSONArray>() {
                            @Override
                            public void onResponse(JSONArray response) {

                                int i;

                                try {

                                    JSONArray jsonArray = new
JSONArray(response.toString());
                                    for (i = 0; i < jsonArray.length();
i++) {
                                        JSONObject jsonObject =
jsonArray.getJSONObject(i);
                                        JSONObject jsonObject1 =
jsonObject.getJSONObject("wst");
                                        String taskType =
jsonObject1.optString("taskType");


                                        String url1 =
"http://192.168.1.3:8080/api/cartasks/bytype?taskType=" + taskType;
                                        int finalI = i;
                                        JsonArrayRequest jsonArrayRequest1
= new JsonArrayRequest(Request.Method.GET, url1, null, new
Response.Listener<JSONArray>() {
                                            @Override
                                            public void
onResponse(JSONArray response) {
                                                try {
                                                    JSONArray jsonArray1 =
new JSONArray(response.toString());
                                                    for (int j = 0; j <
jsonArray1.length(); j++) {
                                                        JSONObject
jsonObject2 = jsonArray1.getJSONObject(j);
                                                        String taskPart =
```

```
jsonObject2.optString("taskPart");

taskArray1.add(taskPart);
                                                    }

                                                    String[] taskArray2 =
new String[taskArray1.size()];
                                                    for (int k = 0; k <
taskArray1.size(); k++) {
                                                        taskArray2[k] =
taskArray1.get(k);
                                                    }
                                                    //Initialize selected
tasks arraylist
                                                    if (finalI ==
jsonArray.length() - 1) {

                                                        selectedTasks = new
boolean[taskArray2.length];

                                                        //Initialize alert
dialog
                                                        AlertDialog.Builder
builder = new AlertDialog.Builder(CarVisitActivity.this);

                                                        //Set title

builder.setTitle(getString(R.string.select_the_tasks_which_were_perfomed));

builder.setCancelable(false);

builder.setMultiChoiceItems(taskArray2, selectedTasks, new
DialogInterface.OnMultiChoiceClickListener() {
                                                            @Override
                                                            public void
onClick(DialogInterface dialog, int which, boolean isChecked) {
                                                                //Check
condition
                                                                if
(isChecked) {
                                                                    if
(!taskList.contains(which)) { //Check if already item on the list

//When checkbox is selected, add position in list

taskList.add(which);
                                                                    }
                                                                } else {
                                                                    if
(!taskList.contains(which)) { //Check if already item on the list (changed)

//When checkbox is unselected, remove position from list

taskList.remove(which);
                                                                    }
                                                                }
```

```
                                                    }
                                                });


builder.setPositiveButton(getString(R.string.dialog_taskList_ok), new
DialogInterface.OnClickListener() {
                                                    @Override
                                                    public void
onClick(DialogInterface dialog, int which) {

//Initialize the string builder

StringBuilder stringBuilder = new StringBuilder();
                                                        for (int j
= 0; j < taskList.size(); j++) {

//Concat array value

stringBuilder.append(taskArray2[taskList.get(j)]);  //Crash here
                                                            //Check
condition
                                                            if (j
!= taskList.size() - 1) {

//When j value not equal to task list size -1 add comma

stringBuilder.append("\n");
                                                            }
                                                        }
                                                        //Set text
on text view

tvTasks.setText(stringBuilder.toString());
                                                    }
                                                });

builder.setNegativeButton(R.string.sb_cancel_tasks, new
DialogInterface.OnClickListener() {
                                                    @Override
                                                    public void
onClick(DialogInterface dialog, int which) {

                                                    }
                                                });


builder.setNeutralButton(getString(R.string.sb_clear_all_tasks), new
DialogInterface.OnClickListener() {
                                                    @Override
                                                    public void
onClick(DialogInterface dialog, int which) {
                                                        for (int j
= 0; j < selectedTasks.length; j++) {

//Remove all selection

selectedTasks[j] = false;

taskList.clear();
```

```
tvTasks.setText("");
                                                                    }
                                                                }
                                                            });
                                                            builder.show();

                                                        }

                                                        //taskArray1 clean

                                                    } catch (JSONException e) {
                                                        e.printStackTrace();
                                                    }
                                                }
                                            }, new Response.ErrorListener() {
                                                @Override
                                                public void
onErrorResponse(VolleyError error) {
                                                    String errormsg;
                                                    errormsg =
VolleyErrorHelper.getMessage(error, CarVisitActivity.this);

Toast.makeText(CarVisitActivity.this, errormsg, Toast.LENGTH_SHORT).show();
                                                }
                                            });
                                            //Add the json request object to
the request queue

RequestSingleton.getInstance(CarVisitActivity.this).addToRequestQueue(jsonA
rrayRequest1);


                                        }


                                    } catch (JSONException e) {
                                        e.printStackTrace();
                                    }

                                }
                            }, new Response.ErrorListener() {
                                @Override
                                public void onErrorResponse(VolleyError error)
{

                                    String errormsg;
                                    errormsg =
VolleyErrorHelper.getMessage(error, CarVisitActivity.this);
                                    Toast.makeText(CarVisitActivity.this,
errormsg, Toast.LENGTH_SHORT).show();

                                }
                            });

                            //Add the json request object to the request queue

RequestSingleton.getInstance(CarVisitActivity.this).addToRequestQueue(jsonA
rrayRequest);
                    }
```

```
                }


            }
        });


    }

    void getSetIntentData() {
        if (getIntent().hasExtra("speedometer")) {
            //Getting data from intent
            speedometer = getIntent().getLongExtra("speedometer", 0);
            //p_date = getIntent().getStringExtra("p_date");
            modelcode = getIntent().getLongExtra("modelcode", 0);
            Brand = getIntent().getStringExtra("brand");
            Model = getIntent().getStringExtra("model");
            Chassis = getIntent().getStringExtra("chassis");
            myCarId = getIntent().getLongExtra("id",0);

        }
    }

    void setAlarm (String title,String date, String time) {
        //inserts the title,date,time into sql lite database

        AlarmManager am = (AlarmManager)
getSystemService(Context.ALARM_SERVICE);
        //assigining alaram manager object to set alaram

        Intent intent = new Intent(getApplicationContext(),
AlarmBroadcast.class);
        intent.putExtra("event", title);
        //sending data to alarm class to create channel and notification
        intent.putExtra("time", date);
        intent.putExtra("date", time);

        PendingIntent pendingIntent =
PendingIntent.getBroadcast(getApplicationContext(), 0, intent,
PendingIntent.FLAG_ONE_SHOT);
        String dateandtime = date + " " + time;
        DateFormat formatter = new SimpleDateFormat("d-M-yyyy hh:mm");
        try {
            Date date1 = formatter.parse(dateandtime);
            am.set(AlarmManager.RTC_WAKEUP, date1.getTime(),
pendingIntent);
            Toast.makeText(getApplicationContext(),
getString(R.string.rem_set_success_msg), Toast.LENGTH_SHORT).show();

        } catch (ParseException e) {
            e.printStackTrace();
        }
    }
}
```

CarVisitAdapter.java

```
package com.example.carbook.servicebook;
```

```
import android.annotation.SuppressLint;
import android.app.Activity;
import android.content.Context;
import android.content.Intent;
import android.database.Cursor;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;
import android.widget.Toast;

import androidx.annotation.NonNull;
import androidx.recyclerview.widget.RecyclerView;

import com.example.carbook.R;
import com.example.carbook.mycars.MyCarModel;
import com.example.carbook.mycars.dbCarManager;

import java.util.ArrayList;

public class CarVisitAdapter extends
RecyclerView.Adapter<CarVisitAdapter.myviewholder> {

    ArrayList<CarVisitModel> dataholderV = new ArrayList<>();
    Context context;
    Activity activity;
    //array list to hold the cars

    public CarVisitAdapter(Activity activity, Context context,
ArrayList<CarVisitModel> dataholder) {
        this.dataholderV = dataholder;
        this.context=context;
        this.activity=activity;
    }


    @NonNull
    @Override
    public CarVisitAdapter.myviewholder onCreateViewHolder(@NonNull
ViewGroup parent, int viewType) {
        View view =
LayoutInflater.from(parent.getContext()).inflate(R.layout.adapter_view_layo
ut, parent, false);
        LayoutInflater layoutInflater = LayoutInflater.from(context);


        //inflates the xml file in recyclerview
        return new CarVisitAdapter.myviewholder(view);
    }

    @Override
    public void onBindViewHolder(@NonNull CarVisitAdapter.myviewholder
holder, @SuppressLint("RecyclerView") int position) {

        //Binds the single car objects to recycler view


        holder.vDate.setText(dataholderV.get(position).getP_date());

holder.vOdometer.setText(String.valueOf(dataholderV.get(position).getSpeedo
meter()));
```

```
        holder.itemView.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Intent intent = new Intent(context,
CarVisitViewActivity.class);
                intent.putExtra("visit_code",
dataholderV.get(position).getVisit_code());
                intent.putExtra("mycar_code",
dataholderV.get(position).getMycar_code());
                intent.putExtra("odometer",
dataholderV.get(position).getSpeedometer());
                intent.putExtra("next_service",
dataholderV.get(position).getNext_service());

intent.putExtra("w_code",dataholderV.get(position).getW_code());
                intent.putExtra("p_date",
String.valueOf(dataholderV.get(position).getP_date()));
                //Starts the new activity
                //Toast.makeText(CarVisitAdapter.this,
String.valueOf(Integer.parseInt(i)) , Toast.LENGTH_SHORT).show();
                activity.startActivity(intent);
            }
        });

    }

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

    class myviewholder extends RecyclerView.ViewHolder {

        TextView vDate, vOdometer;


        public myviewholder(@NonNull View itemView) {
            super(itemView);

            vDate = itemView.findViewById(R.id.textView1);
//holds the reference of the materials to show data in recyclerview
            vOdometer = itemView.findViewById(R.id.textView2);



        }
    }
}
```

CarVisitModel.java

```
package com.example.carbook.servicebook;

public class CarVisitModel {
    int  mycar_code;
    int w_code;

    public CarVisitModel(int visit_code, int mycar_code, int w_code, Long
```

```
speedometer, Long next_service, String p_date) {
        this.visit_code = visit_code;
        this.mycar_code = mycar_code;
        this.w_code = w_code;
        this.speedometer = speedometer;
        this.next_service = next_service;
        this.p_date = p_date;
    }

    public int getVisit_code() {
        return visit_code;
    }

    public void setVisit_code(int visit_code) {
        this.visit_code = visit_code;
    }

    int visit_code;

    public CarVisitModel(Long speedometer, Long next_service, String
p_date) {
        this.speedometer = speedometer;
        this.next_service = next_service;
        this.p_date = p_date;
    }

    Long speedometer, next_service;
    String p_date;


    public int getMycar_code() {
        return mycar_code;
    }

    public void setMycar_code(int mycar_code) {
        this.mycar_code = mycar_code;
    }

    public int getW_code() {
        return w_code;
    }

    public void setW_code(int w_code) {
        this.w_code = w_code;
    }

    public Long getSpeedometer() {
        return speedometer;
    }

    public void setSpeedometer(Long speedometer) {
        this.speedometer = speedometer;
    }

    public Long getNext_service() {
        return next_service;
    }

    public void setNext_service(Long next_service) {
        this.next_service = next_service;
    }
```

```
    public String getP_date() {
        return p_date;
    }

    public void setP_date(String p_date) {
        this.p_date = p_date;
    }

    public CarVisitModel(int mycar_code, Long speedometer, Long
next_service, int w_code, String p_date) {
        this.mycar_code = mycar_code;
        this.w_code = w_code;
        this.speedometer = speedometer;
        this.next_service = next_service;
        this.p_date = p_date;
    }

}
```

CarVisitViewActivity.java

```
package com.example.carbook.servicebook;

import androidx.appcompat.app.AppCompatActivity;

import android.app.AlertDialog;
import android.content.DialogInterface;
import android.content.Intent;
import android.database.Cursor;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;
import android.widget.Toast;

import com.android.volley.Request;
import com.android.volley.Response;
import com.android.volley.VolleyError;
import com.android.volley.toolbox.JsonObjectRequest;
import com.example.carbook.R;
import com.example.carbook.mainactivity.MainActivity;
import com.example.carbook.mainactivity.RequestSingleton;
import com.example.carbook.mycars.MyCarModel;
import com.example.carbook.mycars.MyCarsViewActivity;
import com.example.carbook.mycars.VolleyErrorHelper;
import com.example.carbook.mycars.dbCarManager;
import
com.google.android.material.floatingactionbutton.FloatingActionButton;

import org.json.JSONObject;

import java.util.ArrayList;

public class CarVisitViewActivity extends AppCompatActivity {

    TextView tvDate, tvOdometer, tvNextService, tvWorkshop, tvTasks;
    FloatingActionButton visitDeleteBtn;
    String pDate;
    long odometer,nextService,visitCode;
```

```
    int myCarCode, wCode;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_car_visit_view);

        tvDate = findViewById(R.id.service_date1);
        tvOdometer = findViewById(R.id.sb_odometer);
        tvNextService = findViewById(R.id.sb_next_srv1);

        tvWorkshop = findViewById(R.id.sb_workshops_by_tasks1);
        tvTasks = findViewById(R.id.s_t_by_vs1);

        visitDeleteBtn = findViewById(R.id.my_visit_delete);

        getSetIntentData();

        String url1 = "http://192.168.1.3:8080/api/workshop/" + wCode;
        JsonObjectRequest jsonObjectRequest1 = new
JsonObjectRequest(Request.Method.GET, url1, null, new
Response.Listener<JSONObject>() {
            @Override
            public void onResponse(JSONObject response) {

                String name = response.optString("name");
                tvWorkshop.setText(name);


            }
        }, new Response.ErrorListener() {
            @Override
            public void onErrorResponse(VolleyError error) {
                String errormsg;
                errormsg = VolleyErrorHelper.getMessage(error,
CarVisitViewActivity.this);
                Toast.makeText(CarVisitViewActivity.this, errormsg,
Toast.LENGTH_SHORT).show();
            }
        });

RequestSingleton.getInstance(CarVisitViewActivity.this).addToRequestQueue(j
sonObjectRequest1);

        Cursor cursor = new
dbCarManager(CarVisitViewActivity.this).readOneCarServices(visitCode);
        tvTasks.setText("");
        String task="";
        while (cursor.moveToNext()) {
            String tvBuilder = tvTasks.getText().toString().trim() +"\n";
            task = tvBuilder + cursor.getString(1);
            tvTasks.setText(task);
        }

        visitDeleteBtn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                confirmDialog();
            }
        });
```

```
    }

    void getSetIntentData(){
        if(getIntent().hasExtra("visit_code")){
            //Getting data from intent
            visitCode = getIntent().getIntExtra("visit_code",0);
            myCarCode = getIntent().getIntExtra("mycar_code",0);
            odometer = getIntent().getLongExtra("odometer",0);
            nextService = getIntent().getLongExtra("next_service",0);
            wCode = getIntent().getIntExtra("w_code",0);
            pDate = getIntent().getStringExtra("p_date");

            tvDate.setText(pDate);
            tvOdometer.setText(String.valueOf(odometer));
            tvNextService.setText(String.valueOf(nextService));

        }


    }

    void confirmDialog(){
        AlertDialog.Builder builder = new AlertDialog.Builder(this);
        builder.setMessage(getString(R.string.delete_visit_ask));
        builder.setPositiveButton(R.string.btn_delete_remdialog, new
DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialogInterface, int i) {
                String result1 = new
dbCarManager(CarVisitViewActivity.this).deleteService(visitCode);
                String result2 = new
dbCarManager(CarVisitViewActivity.this).deleteVisit(visitCode);
                Toast.makeText(getApplicationContext(),
getString(R.string.car_deleted), Toast.LENGTH_SHORT).show();
                finish();
                Intent intentBack = new Intent(getApplicationContext(),
MainActivity.class);
                //this intent will be called once the setting alarm
completes
                intentBack.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK |
Intent.FLAG_ACTIVITY_CLEAR_TASK);
                startActivity(intentBack);
            }
        });
        builder.setNegativeButton(R.string.btn_cancel_remdialog, new
DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialogInterface, int i) {
                Toast.makeText(getApplicationContext(),
getString(R.string.del_cancel_remdialog), Toast.LENGTH_SHORT).show();
            }
        });
        builder.create().show();
    }
}
```

ServiceBookActivity.java

```
package com.example.carbook.servicebook;
```

```
import android.content.Intent;
import android.database.Cursor;
import android.os.Bundle;
import android.view.View;
import android.widget.AdapterView;
import android.widget.EditText;
import android.widget.ListView;
import android.widget.TextView;
import android.widget.Toast;

import androidx.appcompat.app.AppCompatActivity;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

import com.example.carbook.R;
import com.example.carbook.mycars.dbCarManager;
import com.example.carbook.reminders.dbManager;
import
com.google.android.material.floatingactionbutton.FloatingActionButton;

import java.util.ArrayList;

public class ServiceBookActivity extends AppCompatActivity {

    String Brand, Model, Chassis;
    long speedometer;
    Long modelcode, myCarId;

    ArrayList<CarVisitModel> dataholder = new ArrayList<CarVisitModel>();

    TextView tvModel,tvBrand,tvModelCodeInvisible;
    EditText etSpeedometer;
    RecyclerView visitList;


    FloatingActionButton addTask;
    CarVisitAdapter adapter;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_service_book);

        visitList = findViewById(R.id.sb_service_task_list);
        tvBrand = findViewById(R.id.mycar_brand_sb);
        tvModel = findViewById(R.id.mycar_model_sb);
        etSpeedometer =findViewById(R.id.sb_speedometer);
        tvModelCodeInvisible = findViewById(R.id.invisible_model_code2);
        addTask = findViewById(R.id.sb_add_task);


        getSetIntentData();

        addTask.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {

                Intent intent = new Intent(ServiceBookActivity.this,
CarVisitActivity.class);
                intent.putExtra("speedometer", speedometer);
                //intent.putExtra("p_date", p_date);
```

```
                intent.putExtra("modelcode", modelcode);
                intent.putExtra("brand", Brand);
                intent.putExtra("model", Model);
                intent.putExtra("chassis", Chassis);
                intent.putExtra("id", myCarId);

                startActivity(intent);
                //Starts the new activity

            }
        });

        Cursor cursor = new
dbCarManager(ServiceBookActivity.this).readModelVisits(myCarId);
        //Cursor cursor = new
dbCarManager(ServiceBookActivity.this).readallVisits();  //Cursor To Load
data From the database
        while (cursor.moveToNext()) {
            CarVisitModel model = new
CarVisitModel(cursor.getInt(0),cursor.getInt(1),cursor.getInt(4) ,
cursor.getLong(2),
                    cursor.getLong(3), cursor.getString(5));

//            CarVisitModel model = new
CarVisitModel(cursor.getInt(0),cursor.getInt(1),cursor.getInt(4) ,
cursor.getLong(3),
//                    cursor.getLong(2), cursor.getString(5));

            dataholder.add(model);
        }
        adapter = new
CarVisitAdapter(ServiceBookActivity.this,getApplicationContext(),dataholder
);
        visitList.setAdapter(adapter);
        visitList.setLayoutManager(new
LinearLayoutManager(ServiceBookActivity.this));



    }

    void getSetIntentData(){
        if(getIntent().hasExtra("chassis")){
            //Getting data from intent
            Chassis = getIntent().getStringExtra("chassis");
            Brand = getIntent().getStringExtra("brand");
            Model = getIntent().getStringExtra("model");
            modelcode = getIntent().getLongExtra("modelcode",0);
            myCarId = getIntent().getLongExtra("id",0);
            tvBrand.setText(Brand);
            tvModel.setText(Model);
            tvModelCodeInvisible.setText(String.valueOf(modelcode));
        }
    }
}
```

ServiceBookAdapter.java

```
package com.example.carbook.servicebook;
```

```
import android.annotation.SuppressLint;
import android.app.Activity;
import android.content.Context;
import android.content.Intent;
import android.database.Cursor;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;
import android.widget.Toast;

import androidx.annotation.NonNull;
import androidx.recyclerview.widget.RecyclerView;

import com.example.carbook.R;
import com.example.carbook.mycars.MyCarModel;
import com.example.carbook.mycars.dbCarManager;

import java.util.ArrayList;

public class ServiceBookAdapter extends
RecyclerView.Adapter<ServiceBookAdapter.myviewholder> {

    ArrayList<MyCarModel> CVdataholder = new ArrayList<>();
    Context context;
    Activity activity;
    //array list to hold the cars

    public ServiceBookAdapter(Activity activity, Context context,
ArrayList<MyCarModel> dataholder) {
        this.CVdataholder = dataholder;
        this.context=context;
        this.activity=activity;
    }


    @NonNull
    @Override
    public ServiceBookAdapter.myviewholder onCreateViewHolder(@NonNull
ViewGroup parent, int viewType) {
        View view =
LayoutInflater.from(parent.getContext()).inflate(R.layout.single_serviceboo
k, parent, false);
        LayoutInflater layoutInflater = LayoutInflater.from(context);


        //inflates the xml file in recyclerview
        return new ServiceBookAdapter.myviewholder(view);
    }

    @Override
    public void onBindViewHolder(@NonNull ServiceBookAdapter.myviewholder
holder, @SuppressLint("RecyclerView") int position) {

        //Binds the single car objects to recycler view
        Cursor result = new
dbCarManager(activity.getApplicationContext()).getMyCarModel((long)
CVdataholder.get(position).getId());
        result.moveToFirst();
        Long modelcode = result.getLong(2);
        String model =  CVdataholder.get(position).getMyC_model();
```

```
//result.getString(0);
        String year =  CVdataholder.get(position).getMyC_year();
//result.getString(2);
        String chassis = CVdataholder.get(position).getChassis_number();
        String brand = CVdataholder.get(position).getMyC_brand();
        Long myCarId = CVdataholder.get(position).getId();


        holder.sModel.setText(String.valueOf(model));
        holder.sDate.setText(String.valueOf(year));
        holder.sChassis.setText(String.valueOf(chassis));

        holder.itemView.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Intent intent = new Intent(context,
ServiceBookActivity.class);
                intent.putExtra("chassis", String.valueOf(chassis));
                intent.putExtra("model", String.valueOf(model));
                intent.putExtra("brand", String.valueOf(brand));
                intent.putExtra("modelcode", modelcode);
                intent.putExtra("id", myCarId);
                //Starts the new activity to edit car
                activity.startActivity(intent);
            }
        });

    }

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

    class myviewholder extends RecyclerView.ViewHolder {

        TextView sModel, sDate, sChassis;


        public myviewholder(@NonNull View itemView) {
            super(itemView);

            sModel = itemView.findViewById(R.id.tv_model_visit);
//holds the reference of the materials to show data in recyclerview
            sDate = itemView.findViewById(R.id.tv_date_visit);
            sChassis = itemView.findViewById(R.id.tv_klm);

        }
    }
}
```

ServiceBookFragmenet.java

```
package com.example.carbook.servicebook;

import android.database.Cursor;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
```

```
import android.widget.TextView;

import androidx.annotation.Nullable;
import androidx.appcompat.app.AppCompatActivity;
import androidx.fragment.app.Fragment;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

import com.example.carbook.R;
import com.example.carbook.mycars.MyCarModel;
import com.example.carbook.mycars.dbCarManager;

import java.util.ArrayList;

public class ServiceBookFragment extends Fragment {

    RecyclerView serviceBookRecyclerview;
    ArrayList<MyCarModel> dataholder = new ArrayList<MyCarModel>();
    ServiceBookAdapter adapter;

    TextView tvEmpty;

    @Nullable
    @Override
    public View onCreateView(LayoutInflater inflater, @Nullable  ViewGroup
container, @Nullable  Bundle savedInstanceState) {

        View view = inflater.inflate(R.layout.fragment_servicebook,
container, false);


((AppCompatActivity)getActivity()).getSupportActionBar().setTitle(getString
(R.string.service_book));

        serviceBookRecyclerview =
view.findViewById(R.id.service_book_recycler);
        serviceBookRecyclerview.setLayoutManager(new
LinearLayoutManager(getContext()));
        tvEmpty = view.findViewById(R.id.empty2);

        Cursor cursor1 = new dbCarManager(getContext()).carIsEmpty();
        cursor1.moveToFirst();
        int count = cursor1.getInt(0);
        if(count != 0) {
            tvEmpty.setVisibility(View.INVISIBLE);
        } else {tvEmpty.setVisibility(View.VISIBLE);}


        Cursor cursor = new dbCarManager(getContext()).readallcars();
//Cursor To Load data From the database
        while (cursor.moveToNext()) {
            MyCarModel model = new
MyCarModel(cursor.getLong(0),cursor.getString(1), cursor.getString(2),
cursor.getString(3),
                    cursor.getString(4), cursor.getString(5),
cursor.getString(6), cursor.getString(7),
                    cursor.getString(8), cursor.getString(9),
cursor.getString(10), cursor.getString(11), cursor.getString(12),
cursor.getLong(13));
            dataholder.add(model);
        }
```

```
        adapter = new
ServiceBookAdapter(getActivity(),getContext(),dataholder);
        serviceBookRecyclerview.setAdapter(adapter); //Binds the adapter
with recyclerview
        serviceBookRecyclerview.setLayoutManager(new
LinearLayoutManager(getActivity()));

        return view;
    }
}
```

VisitScheduleModel.java

```
package com.example.carbook.servicebook;

public class VisitScheduleModel {
    Long sCode,modelCode,taskCode,kilometers;
    int months;

    public Long getsCode() {
        return sCode;
    }

    public void setsCode(Long sCode) {
        this.sCode = sCode;
    }

    public Long getModelCode() {
        return modelCode;
    }

    public void setModelCode(Long modelCode) {
        this.modelCode = modelCode;
    }

    public Long getTaskCode() {
        return taskCode;
    }

    public void setTaskCode(Long taskCode) {
        this.taskCode = taskCode;
    }

    public Long getKilometers() {
        return kilometers;
    }

    public void setKilometers(Long kilometers) {
        this.kilometers = kilometers;
    }

    public int getMonths() {
        return months;
    }

    public void setMonths(Integer months) {
        this.months = months;
    }
```

```
    public VisitScheduleModel(Long sCode, Long modelCode, Long taskCode,
Long kilometers, int months) {
        this.sCode = sCode;
        this.modelCode = modelCode;
        this.taskCode = taskCode;
        this.kilometers = kilometers;
        this.months = months;
    }
}
```

SettingsFragment.java

```
package com.example.carbook.settings;


import android.content.Context;
import android.content.SharedPreferences;
import android.media.MediaPlayer;
import android.os.Bundle;
import android.widget.Toast;

import androidx.appcompat.app.ActionBar;
import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.app.AppCompatDelegate;
import androidx.preference.ListPreference;
import androidx.preference.Preference;
import androidx.preference.PreferenceFragmentCompat;
import androidx.preference.PreferenceManager;
import androidx.preference.SwitchPreferenceCompat;

import com.example.carbook.R;


public class SettingsFragment extends PreferenceFragmentCompat {


    MediaPlayer player;



    @Override
    public void onCreatePreferences(Bundle savedInstanceState, String
rootKey) {
        // Load the preferences from an XML resource
        setPreferencesFromResource(R.xml.fragment_settings, rootKey);


        loadSettings();



    }

    private void loadSettings() {
        SharedPreferences sp =
PreferenceManager.getDefaultSharedPreferences(getContext());
        SharedPreferences n =
getActivity().getSharedPreferences("Notification_prf",
Context.MODE_PRIVATE);
```

```
        SharedPreferences.Editor e = n.edit();
        //This the preference for the music
        Preference pMusic =
getPreferenceManager().findPreference("switch_music");
        //This loads the music setting


        pMusic.setOnPreferenceClickListener(new
Preference.OnPreferenceClickListener() {
            @Override
            public boolean onPreferenceClick(Preference preference) {
                if (player==null){
                    mThread mthread = new mThread();
                    mthread.start();
                }

                else  {
                    // sMusic.setChecked(false);
                    player.release();
                    player = null;
                }
                return false;
            }
        });

        //This loads the theme setting
        String spTheme = sp.getString("theme", "false");
        if ("1".equals(spTheme)){

AppCompatDelegate.setDefaultNightMode(AppCompatDelegate.MODE_NIGHT_NO);
        }
        else if ("2".equals(spTheme)){

AppCompatDelegate.setDefaultNightMode(AppCompatDelegate.MODE_NIGHT_YES);
        }
        else if ("3".equals(spTheme)){

AppCompatDelegate.setDefaultNightMode(AppCompatDelegate.MODE_NIGHT_FOLLOW_S
YSTEM);
        }

        //This the preference for the theme
        ListPreference lTheme = (ListPreference)
getPreferenceManager().findPreference("theme");
        lTheme.setOnPreferenceChangeListener(new
Preference.OnPreferenceChangeListener() {
            @Override
            public boolean onPreferenceChange(Preference preference, Object
newValue) {
                String items = (String)newValue;
                if(preference.getKey().equals("theme")) {
                    switch (items){
                        case "1":

AppCompatDelegate.setDefaultNightMode(AppCompatDelegate.MODE_NIGHT_NO);
                            break;
                        case "2":

AppCompatDelegate.setDefaultNightMode(AppCompatDelegate.MODE_NIGHT_YES);
                            break;
                        case "3":
```

```
AppCompatDelegate.setDefaultNightMode(AppCompatDelegate.MODE_NIGHT_FOLLOW_S
YSTEM);
                                break;
                        }
                    }
                    return true;
                }
            });
            lTheme.setNegativeButtonText(getString(R.string.pref_list_cancel));

            //This loads the language setting


            //This the preference for the language
            ListPreference lLanguage = (ListPreference)
getPreferenceManager().findPreference("language");
            LanguageManager lang = new LanguageManager(getContext());
            lLanguage.setOnPreferenceChangeListener(new
Preference.OnPreferenceChangeListener() {
                @Override
                public boolean onPreferenceChange(Preference preference, Object
newValue) {
                    String items = (String)newValue;
                    if(preference.getKey().equals("language")) {
                        switch (items) {
                            case "1":
                                lang.setLocal("grc");
                                getActivity().recreate();
                                break;
                            case "2":
                                lang.setLocal("en");
                                getActivity().recreate();
                                break;
                        }
                    }
                    return true;
                }
            });

lLanguage.setNegativeButtonText(getString(R.string.pref_list_cancel));



            SwitchPreferenceCompat pNotif = (SwitchPreferenceCompat)
getPreferenceManager().findPreference("switch_notifications");
            pNotif.setOnPreferenceChangeListener(new
Preference.OnPreferenceChangeListener() {
                @Override
                public boolean onPreferenceChange(Preference preference, Object
newValue) {
                    if (pNotif.isChecked()) {
                        Toast.makeText(getContext(),
getString(R.string.preference_msg_disabled_notif),
Toast.LENGTH_SHORT).show();
                        e.putBoolean("notif",false);
                        e.apply();
                    } else if (!pNotif.isChecked()){
                        Toast.makeText(getContext(),
getString(R.string.preference_msg_enabled_notif),
Toast.LENGTH_SHORT).show();
```

```
                    e.putBoolean("notif",true);
                    e.apply();
                }

                return true;
            }
        });


    }



    class mThread extends Thread {
        @Override
        public void run() {
            if(player==null) {
                player = MediaPlayer.create(getContext(), R.raw.song);
                player.start();
            }
            }
        }

    @Override
    public void onResume() {
        loadSettings();
        super.onResume();
    }




}
```

LanguageManager.java

```
package com.example.carbook.settings;

import android.content.Context;
import android.content.SharedPreferences;
import android.content.res.Configuration;
import android.content.res.Resources;

import java.util.Locale;

public class LanguageManager {
    private Context ct;
    private SharedPreferences sharedPreferences;
    public LanguageManager(Context ctx){
            ct=ctx;
            sharedPreferences =
ct.getSharedPreferences("Lang",Context.MODE_PRIVATE);
    }
    public void setLocal(String lang){
        Locale locale = new Locale(lang);
        Locale.setDefault(locale);
        Resources resources = ct.getResources();
        Configuration config = resources.getConfiguration();
        config.locale = locale;
```

```
resources.updateConfiguration(config,resources.getDisplayMetrics());
        setLang(lang);

    }

    public String getLang(){
        return sharedPreferences.getString("Lang","grc");
    }

    public void setLang(String lang){
        //Save date to shared preferences.
        SharedPreferences.Editor editor = sharedPreferences.edit();
        editor.putString("Lang",lang);
        editor.apply();
    }

}
```

MyAppCompat.java

```
package com.example.carbook.settings;

import android.os.Bundle;

import androidx.annotation.Nullable;
import androidx.appcompat.app.AppCompatActivity;

public class MyAppCompat extends AppCompatActivity {
    @Override
    protected void onCreate(@Nullable @org.jetbrains.annotations.Nullable
Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        LanguageManager languageManager = new LanguageManager(this);
        languageManager.setLocal(languageManager.getLang());
    }
}
```

MapFragment.java

```
package com.example.carbook;

import static com.example.carbook.R.id.sv_location;

import android.Manifest;
import android.content.pm.PackageManager;
import android.location.Address;
import android.location.Geocoder;
import android.location.Location;
import android.location.LocationListener;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Toast;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.widget.SearchView;
import androidx.core.content.ContextCompat;
```

```
import androidx.fragment.app.Fragment;

import com.android.volley.Request;
import com.android.volley.Response;
import com.android.volley.VolleyError;
import com.android.volley.toolbox.JsonArrayRequest;
import com.example.carbook.mainactivity.RequestSingleton;
import com.example.carbook.mycars.VolleyErrorHelper;
import com.google.android.gms.common.ConnectionResult;
import com.google.android.gms.common.api.GoogleApiClient;
import com.google.android.gms.location.FusedLocationProviderClient;
import com.google.android.gms.location.LocationServices;
import com.google.android.gms.maps.CameraUpdateFactory;
import com.google.android.gms.maps.GoogleMap;
import com.google.android.gms.maps.OnMapReadyCallback;
import com.google.android.gms.maps.SupportMapFragment;
import com.google.android.gms.maps.model.LatLng;
import com.google.android.gms.maps.model.MarkerOptions;

import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

import java.io.IOException;
import java.util.List;


public class MapFragment extends Fragment implements
OnMapReadyCallback,GoogleApiClient.ConnectionCallbacks,
        GoogleApiClient.OnConnectionFailedListener,
        LocationListener {

    GoogleMap mMap;
    FusedLocationProviderClient client;
    GoogleApiClient mGoogleApiClient;
    SearchView searchView;
    LatLng Athens;



    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
Bundle savedInstanceState) {

        View view = inflater.inflate(R.layout.fragment_map, container,
false);

        setHasOptionsMenu(true);

        //Initialize map fragment
        searchView = (SearchView) view.findViewById(sv_location);
        searchView.setQueryHint(getString(R.string.sv_txt));
        SupportMapFragment supportMapFragment = (SupportMapFragment)

getChildFragmentManager().findFragmentById(R.id.google_map);

        //Initialize fused location
        client =
LocationServices.getFusedLocationProviderClient(requireContext());
```

```
        //Create a search view listener.
        searchView.setOnQueryTextListener(new
SearchView.OnQueryTextListener() {
            @Override
            public boolean onQueryTextSubmit(String query) {
                String location = searchView.getQuery().toString();
                List<Address> addressList=null;
                if(location != null || !location.equals((""))){
                    Geocoder geocoder = new Geocoder(getContext());
                    try {
                        addressList=geocoder.getFromLocationName(location,
1);
                    } catch (IOException e) {
                        e.printStackTrace();
                    }
                    if(!addressList.isEmpty()) {
                        Address address = addressList.get(0);
                        LatLng latLng = new LatLng(address.getLatitude(),
address.getLongitude());
                        mMap.addMarker(new
MarkerOptions().position(latLng).title(location));

mMap.animateCamera(CameraUpdateFactory.newLatLngZoom(latLng, 15));
                    } else {
                        Toast.makeText(getContext(),
R.string.loc_not_found, Toast.LENGTH_SHORT).show();
                    }
                }
                return false;
            }

            @Override
            public boolean onQueryTextChange(String newText) {
                return false;
            }
        });


        //Async map
        assert supportMapFragment != null;
        supportMapFragment.getMapAsync(googleMap -> {
            //When map is loaded
            mMap = googleMap;

            mMap.getUiSettings().setZoomControlsEnabled(true);
            mMap.getUiSettings().setZoomGesturesEnabled(true);
            mMap.getUiSettings().setCompassEnabled(true);
            mMap.setTrafficEnabled(true);

            Athens = new LatLng(38.06386077178782, 23.765054479058882);
            mMap.animateCamera(CameraUpdateFactory.newLatLngZoom(Athens,
13));



            //mMap.setMapType(GoogleMap.MAP_TYPE_SATELLITE);

            //Initialize Google Play Services
            if (ContextCompat.checkSelfPermission(getContext(),
```

```
                    Manifest.permission.ACCESS_FINE_LOCATION)
                    == PackageManager.PERMISSION_GRANTED) {
                buildGoogleApiClient();
                mMap.setMyLocationEnabled(true);
            }
            googleMap.setOnMapClickListener(latLng -> {

                //When clicked on map
                //Initialize marker options
                MarkerOptions markerOptions = new MarkerOptions();
                //Set position of the marker
                markerOptions.position(latLng);
                //Set title of marker
                markerOptions.title(getString(R.string.pin_text1) +" "+
latLng.latitude + "  :  " + latLng.longitude);
                //Remove all markers
                googleMap.clear();
                //Animating to zoom the marker
                googleMap.animateCamera(CameraUpdateFactory.newLatLngZoom(
                        latLng, 12));
                //Add marker on map
                googleMap.addMarker(markerOptions);

            });
        });



        String url = "http://192.168.1.3:8080/api/workshops"; //For android
phone

        JsonArrayRequest jsonArrayRequest = new
JsonArrayRequest(Request.Method.GET, url, null, new
Response.Listener<JSONArray>() {
            @Override
            public void onResponse(JSONArray response) {

                try {
                    JSONArray jsonArray = new
JSONArray(response.toString());
                    for(int i=0; i<jsonArray.length(); i++){
                        JSONObject jsonObject = jsonArray.getJSONObject(i);
                        String name = jsonObject.optString("name");
                        String coordinates =
jsonObject.optString("coordinates");


                        String[] latlong =  coordinates.split(" ");
                        double latitude = Double.parseDouble(latlong[0]);
                        double longitude = Double.parseDouble(latlong[1]);
                        // Add a marker in the location and move the camera
                        LatLng location = new LatLng(latitude, longitude);
                        if(mMap!=null){mMap.addMarker(new
MarkerOptions().position(location).title(name));}



                    }
```

```
                } catch (JSONException e) {
                    e.printStackTrace();
                }

            }
        }, new Response.ErrorListener() {
            @Override
            public void onErrorResponse(VolleyError error) {

                String errormsg;
                errormsg= VolleyErrorHelper.getMessage(error,getContext());
                Toast.makeText(getContext(), errormsg,
Toast.LENGTH_SHORT).show();

            }
        });

        //Add the json request object to the request queue

RequestSingleton.getInstance(getContext()).addToRequestQueue(jsonArrayReque
st);


        ((AppCompatActivity)
getActivity()).getSupportActionBar().setTitle(R.string.map);

        return view;
    }




    @Override
    public void onMapReady(GoogleMap googleMap) {

    }

    protected synchronized void buildGoogleApiClient() {
        mGoogleApiClient = new GoogleApiClient.Builder(getContext())
                .addConnectionCallbacks(this)
                .addOnConnectionFailedListener(this)
                .addApi(LocationServices.API)
                .build();
        mGoogleApiClient.connect();
    }




    @Override
    public void onLocationChanged(@NonNull Location location) {

    }

    @Override
    public void onConnected(@Nullable @org.jetbrains.annotations.Nullable
Bundle bundle) {

    }

    @Override
```

```
    public void onConnectionSuspended(int i) {

    }

    @Override
    public void onConnectionFailed(@NonNull ConnectionResult
connectionResult) {

    }

}
```

Παράρτημα Β: Κώδικας του Web Service της εφαρμογής CarBook σε Java

CarBrands.java

```
package carbook.webservice.CarBookAPI.model;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Table;

@Entity
@Table(name = "CarBrands")

public class CarBrands {

	  @Id
	@GeneratedValue(strategy = GenerationType.IDENTITY)
	@Column(name="brandCode")
	  private Long id;

	  @Column(name="name")
	private String name;

	@Column(name="country")
	private String country;

	 public String getName() {
		// TODO Auto-generated method stub
		return name;
	 }
	 public String getCountry() {
		// TODO Auto-generated method stub
		return country;
	 }
	 public void setName(String name2) {
		// TODO Auto-generated method stub
		name=name2;

	 }
	 public void setCountry(String country2) {
		// TODO Auto-generated method stub
		country=country2;

	 }
	 public Long getId() {
		return id;
	 }
	 public void setId(Long id) {
		this.id = id;
	 }
}
```

CarBrandsRepository.java

```
package carbook.webservice.CarBookAPI.repository;

import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

import carbook.webservice.CarBookAPI.model.CarBrands;

@Repository
public interface CarBrandsRepository extends JpaRepository<CarBrands, Long>
{

	CarBrands findByName (String name);

}
```

CarBrandsService.java

```
package carbook.webservice.CarBookAPI.service;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import carbook.webservice.CarBookAPI.model.CarBrands;
import carbook.webservice.CarBookAPI.repository.CarBrandsRepository;
import java.util.List;

@Service
public class CarBrandsService {
	@Autowired
	CarBrandsRepository carbRepository;


	// READ
	public List<CarBrands> getCarBrands(){
		return carbRepository.findAll();
	}

	//READ ONE
		public CarBrands getCarBrandByBrandCode(Long brandCode){
			return carbRepository.findById(brandCode).get();
		}

	//READ MODELS BY MODEL NAME & YEAR
		public CarBrands findByName(String name) {
			return carbRepository.findByName(name);
		}



}
```

CarBrandsController.java

```
package carbook.webservice.CarBookAPI.controller;

import java.util.List;
```

```
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.ResponseStatus;
import org.springframework.web.bind.annotation.RestController;

import carbook.webservice.CarBookAPI.model.CarBrands;
import carbook.webservice.CarBookAPI.service.CarBrandsService;

@RestController
@RequestMapping("/api")
public class CarBrandsController {
	@Autowired
	CarBrandsService carbService;


	@RequestMapping(value="/carbrands", method=RequestMethod.GET)
	public List<CarBrands> readeCarBrands() {
		return carbService.getCarBrands();
	}


	//Get one carbrand
	@RequestMapping(value="/carbrand/{brandCode}",
method=RequestMethod.GET)
	public CarBrands readeCarBrandById(@PathVariable(value = "brandCode")
Long id) {
		return carbService.getCarBrandByBrandCode(id);
	}

	//brand by name
	//http://localhost:8080/api/byname/NISSAN
    @GetMapping("/byname/{name}")
    @ResponseStatus(HttpStatus.OK)
    public CarBrands getBrandByName(@PathVariable(value = "name") final
String name) {
        return carbService.findByName(name);
    }


}
```

CarModels.java

```
package carbook.webservice.CarBookAPI.model;


import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Table;

@Entity
@Table(name = "CarModels")
```

```
public class CarModels {

      @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name="modelCode")
       private Long id;

      @Column(name="brand")
    private Long brand;

      @Column(name="model")
    private String model;

      @Column(name="year")
    private String year;

      @Column(name="cubic")
    private Long cubic;

      @Column(name="horsePower")
    private Long horsePower;

      @Column(name="fuelType")
    private String fuelType;



      public Long getId() {
            return id;
      }

      public void setId(Long id) {
            this.id = id;
      }

      public Long getBrand() {
            return brand;
      }

      public void setBrand(Long brand) {
            this.brand = brand;
      }

      public String getModel() {
            return model;
      }

      public void setModel(String model) {
            this.model = model;
      }

      public String getYear() {
            return year;
      }

      public void setYear(String year) {
            this.year = year;
      }

      public Long getCubic() {
            return cubic;
```

```
	}

	public void setCubic(Long cubic) {
		this.cubic = cubic;
	}

	public Long getHorsePower() {
		return horsePower;
	}

	public void setHorsePower(Long horsePower) {
		this.horsePower = horsePower;
	}

	public String getFuelType() {
		return fuelType;
	}

	public void setFuelType(String fuelType) {
		this.fuelType = fuelType;
	}

}
```

CarModelsRepository.java

```
package carbook.webservice.CarBookAPI.repository;

import java.util.List;

import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

import carbook.webservice.CarBookAPI.model.CarModels;

@Repository
public interface CarModelsRepository extends JpaRepository <CarModels,
Long> {

	List<CarModels> findByBrand (Long brand);
	CarModels findByModelAndYear (String model, String year);

}
```

CarModelsService.java

```
package carbook.webservice.CarBookAPI.service;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import carbook.webservice.CarBookAPI.model.CarModels;
import carbook.webservice.CarBookAPI.repository.CarModelsRepository;

import java.util.List;

@Service


public class CarModelsService {
```

```
	@Autowired
	CarModelsRepository carmRepository;


		//READ MODELS BY BRAND CODE
		public List<CarModels> findByBrand(Long brand){

			return carmRepository.findByBrand(brand);
		}

		//READ MODELS BY MODEL NAME & YEAR
		public CarModels findByModelAndYear(String model, String year)
{
			return carmRepository.findByModelAndYear(model, year);
		}



}
```

CarModelsController.java

```
package carbook.webservice.CarBookAPI.controller;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.ResponseStatus;
import org.springframework.web.bind.annotation.RestController;

import carbook.webservice.CarBookAPI.model.CarModels;
import carbook.webservice.CarBookAPI.service.CarModelsService;


@RestController
@RequestMapping("/api")

public class CarModelsController {

	@Autowired
	CarModelsService carmService;


	//Get carmodels by brand code
	//http://localhost:8080/api/bybrand?brand=1
    @GetMapping("/bybrand")
    @ResponseStatus(HttpStatus.OK)
    public List<CarModels> getCarModelsByBrand(@RequestParam(name =
"brand") final Long brand) {
        return carmService.findByBrand(brand);
    }

    //model by modelCode and year
```

```
  //Get carmodels by model name and year
      //http://localhost:8080/api/bymodelyear/Micra/2016-2021
      @GetMapping("/bymodelyear/{model}/{year}")
      @ResponseStatus(HttpStatus.OK)
      public CarModels getmodelByModelYear(@PathVariable(name = "model")
final String model, @PathVariable(name = "year") final String year) {
        return carmService.findByModelAndYear(model,year);

      }


}
```

CarTasks.java

```
package carbook.webservice.CarBookAPI.model;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Table;

@Entity
@Table(name = "CarTasks")

public class CarTasks {

       @Id
     @GeneratedValue(strategy = GenerationType.IDENTITY)
     @Column(name="taskCode")
       private Long id;

       @Column(name="taskPart")
     private String taskPart;

     @Column(name="taskType")
     private String taskType;



      public Long getId() {
            return id;
      }

      public void setId(Long id) {
            this.id = id;
      }

      public String getTaskPart() {
            return taskPart;
      }

      public void setTaskPart(String taskPart) {
            this.taskPart = taskPart;
      }

      public String getTaskType() {
```

```
		return taskType;
	}

	public void setTaskType(String taskType) {
		this.taskType = taskType;
	}




}
```

CarTasksRepository.java

```
package carbook.webservice.CarBookAPI.repository;

import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

import java.util.List;
import carbook.webservice.CarBookAPI.model.CarTasks;

@Repository
public interface CarTasksRepository extends JpaRepository <CarTasks, Long>
{

	List<CarTasks> findByTaskType(String taskType);

}
```

CarTasksService.java

```
package carbook.webservice.CarBookAPI.service;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import carbook.webservice.CarBookAPI.model.CarTasks;
import carbook.webservice.CarBookAPI.repository.CarTasksRepository;
import java.util.List;

@Service

public class CarTasksService {

	@Autowired
	CarTasksRepository cartRepository;


		// READ one cartask
		public CarTasks getCarTaskByTaskCode(Long taskCode){
			return cartRepository.findById(taskCode).get();
		}

		//Get by taskType
		public List<CarTasks> findByTaskType(final String taskType) {
		  return cartRepository.findByTaskType(taskType);
		}
```

```
}
```

CarTasksController.java

```
package carbook.webservice.CarBookAPI.controller;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.ResponseStatus;
import org.springframework.web.bind.annotation.RestController;

import carbook.webservice.CarBookAPI.model.CarTasks;
import carbook.webservice.CarBookAPI.service.CarTasksService;


@RestController
@RequestMapping("/api")
public class CarTasksController {

	//http://localhost:8080/api/cartasks

	@Autowired
	CarTasksService cartService;


	//Get by taskCode

	@RequestMapping(value="/cartask/{taskCode}",
method=RequestMethod.GET)
	public CarTasks readeCarTasksById(@PathVariable(value = "taskCode")
Long taskCode) {
		return cartService.getCarTaskByTaskCode(taskCode);
	}

	//Get by taskType
	//http://localhost:8080/api/cartasks/bytype?taskType=brakes
    @GetMapping("/cartasks/bytype")
    @ResponseStatus(HttpStatus.OK)
    public List<CarTasks> getCarTasksByTaskType(@RequestParam(name =
"taskType") final String taskType) {
        return cartService.findByTaskType(taskType);
    }


}
```

VisitSchedule.java

```
package carbook.webservice.CarBookAPI.model;
import javax.persistence.Column;
```

```
import javax.persistence.EmbeddedId;
import javax.persistence.Entity;
import javax.persistence.Table;
import org.springframework.stereotype.Component;

@Entity
@Table(name = "VisitSchedule")

@Component
public class VisitSchedule {
	@EmbeddedId
	VisitScheduleId VS;


	@Column(name="taskCode")
	 private Long taskCode;

	@Column(name="kilometers")
	 private Long kilometers;

	@Column(name="months")
	 private int months;

	public Long getTaskCode() {
		return taskCode;
	}

	public void setTaskCode(Long taskCode) {
		this.taskCode = taskCode;
	}

	public Long getKilometers() {
		return kilometers;
	}

	public void setKilometers(Long kilometers) {
		this.kilometers = kilometers;
	}

	public int getMonths() {
		return months;
	}

	public void setMonths(int months) {
		this.months = months;
	}

	public VisitScheduleId getVS() {
		return VS;
	}

	public void setVS(VisitScheduleId vS) {
		VS = vS;
	}
}
```

VisitScheduleId.java

```
package carbook.webservice.CarBookAPI.model;
```

```
import java.io.Serializable;
import java.util.Objects;

import javax.persistence.Column;
import javax.persistence.Embeddable;


@Embeddable
public class VisitScheduleId implements Serializable {


	public VisitScheduleId() {

	}
	public VisitScheduleId(Long sCode, Long modelCode) {
		super();
		this.sCode = sCode;
		this.modelCode = modelCode;
	}


	private static final long serialVersionUID = 1L;

	@Column(name = "sCode", nullable = false)
    Long sCode;
    @Column(name = "modelCode", nullable = false)
    Long modelCode;



	public Long getsCode() {
		return sCode;
	}
	public void setsCode(Long sCode) {
		this.sCode = sCode;
	}
	public Long getModelCode() {
		return modelCode;
	}
	public void setModelCode(Long modelCode) {
		this.modelCode = modelCode;
	}


	@Override
	public int hashCode() {
		return Objects.hash(modelCode, sCode);
	}


	@Override
	public boolean equals(Object obj) {
		if (this == obj)
			return true;
		if (obj == null)
			return false;
		if (getClass() != obj.getClass())
			return false;
```

```
            VisitScheduleId other = (VisitScheduleId) obj;
            return Objects.equals(modelCode, other.modelCode) &&
Objects.equals(sCode, other.sCode);
      }



}
```

VisitScheduleRepository.java

```
package carbook.webservice.CarBookAPI.repository;

import carbook.webservice.CarBookAPI.model.VisitSchedule;
import carbook.webservice.CarBookAPI.model.VisitScheduleId;

import java.util.List;

import org.springframework.data.repository.CrudRepository;
import org.springframework.stereotype.Repository;


@Repository
public interface VisitScheduleRepository extends CrudRepository
<VisitSchedule, VisitScheduleId> {


      List<VisitSchedule> findByVSModelCode (Long modelCode);

}
```

VisitScheduleService.java

```
package carbook.webservice.CarBookAPI.service;

import carbook.webservice.CarBookAPI.exception.EntityNotFound;
import carbook.webservice.CarBookAPI.model.VisitSchedule;
import carbook.webservice.CarBookAPI.model.VisitScheduleId;
import carbook.webservice.CarBookAPI.repository.VisitScheduleRepository;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.util.List;
import java.util.stream.Collectors;
import java.util.stream.StreamSupport;



@Service
public class VisitScheduleService {

      @Autowired
      VisitScheduleRepository VSrepository;

      public List<VisitSchedule> getAll() {
        final Iterable<VisitSchedule> iterable = VSrepository.findAll();
        return StreamSupport.stream(iterable.spliterator(),
false).collect(Collectors.toList());
    }
```

```
    //find a book using the composite key - id and isbn id
    public VisitSchedule findByCId(final Long sCode, final Long modelCode)
            throws EntityNotFound {
        final VisitScheduleId identity = new VisitScheduleId(sCode,
modelCode);
        return
VSrepository.findById(identity).orElseThrow(EntityNotFound::new);
    }


    //Find schedule for a specific model (by modelCode)
    public List<VisitSchedule> findByVSModelCode(final Long modelCode) {
        return VSrepository.findByVSModelCode(modelCode);
    }

}
```

VisitScheduleController.java

```
package carbook.webservice.CarBookAPI.controller;
import carbook.webservice.CarBookAPI.exception.EntityNotFound;
import carbook.webservice.CarBookAPI.model.VisitSchedule;
import carbook.webservice.CarBookAPI.service.VisitScheduleService;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.ResponseStatus;
import org.springframework.web.bind.annotation.RestController;
import org.springframework.http.HttpStatus;

import java.util.List;



@RestController
@RequestMapping("/api")
public class VisitScheduleController {

      @Autowired
      VisitScheduleService VSservice;


      //http://localhost:8080/api/schedule
      @GetMapping("/schedule")
    @ResponseStatus(HttpStatus.OK)
    public List<VisitSchedule> getVisitSchedule() {
        return VSservice.getAll();
    }

    // composite key
    //http://localhost:8080/api/modelschedule?sCode=1&modelCode=1
    @GetMapping("/modelschedule")
    @ResponseStatus(HttpStatus.OK)
    public VisitSchedule getVisitScheduleByCompositeKey(@RequestParam(name
= "sCode") final Long sCode,
                                                        @RequestParam(name = "modelCode")
final Long modelCode)
```

```
            throws EntityNotFound {
        return VSservice.findByCId(sCode, modelCode);
    }

    //Used
  //http://localhost:8080/api/bymodel?modelCode=1   localhost: 192.168.1.12
    @GetMapping("/bymodel")
    @ResponseStatus(HttpStatus.OK)
    public List<VisitSchedule> getVisitScheduleByModel(@RequestParam(name =
"modelCode") final Long modelCode) {
        return VSservice.findByVSModelCode(modelCode);
    }



}
```

WorkShops.java

```
package carbook.webservice.CarBookAPI.model;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Table;

@Entity
@Table(name = "WorkShops")

public class WorkShops {

       @Id
     @GeneratedValue(strategy = GenerationType.IDENTITY)
     @Column(name="codeW")
       private Long id;

       @Column(name="name")
     private String name;

       @Column(name="area")
     private String area;

       @Column(name="address")
     private String address;

       @Column(name="coordinates")
     private String coordinates;

       @Column(name="telephone")
     private String telephone;



      public Long getId() {
            return id;
      }

      public void setId(Long id) {
```

```
		this.id = id;
	}

	public String getName() {
		return name;
	}

	public void setName(String name) {
		this.name = name;
	}

	public String getArea() {
		return area;
	}

	public void setArea(String area) {
		this.area = area;
	}

	public String getAddress() {
		return address;
	}

	public void setAddress(String address) {
		this.address = address;
	}

	public String getCoordinates() {
		return coordinates;
	}

	public void setCoordinates(String coordinates) {
		this.coordinates = coordinates;
	}

	public String getTelephone() {
		return telephone;
	}

	public void setTelephone(String telephone) {
		this.telephone = telephone;
	}

}
```

WorkShopsRepository.java

```
package carbook.webservice.CarBookAPI.repository;


import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

import carbook.webservice.CarBookAPI.model.WorkShops;

@Repository
public interface WorkShopsRepository extends JpaRepository <WorkShops,
Long> {

}
```

WorkShopsService.java

```
package carbook.webservice.CarBookAPI.service;


import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import carbook.webservice.CarBookAPI.model.WorkShops;
import carbook.webservice.CarBookAPI.repository.WorkShopsRepository;
import java.util.List;

@Service

public class WorkShopsService {

	@Autowired
	WorkShopsRepository WSRepository;


		// READ
		public List<WorkShops> getWorkShops(){
			return WSRepository.findAll();
		}

		//READ ONE
				public WorkShops getWorkShopByCodeW(Long CodeW){
					return WSRepository.findById(CodeW).get();
				}


}
```

WorkShopsController.java

```
package carbook.webservice.CarBookAPI.controller;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RestController;

import carbook.webservice.CarBookAPI.model.WorkShops;
import carbook.webservice.CarBookAPI.service.WorkShopsService;


@RestController
@RequestMapping("/api")
public class WorkShopsController {

	@Autowired
	WorkShopsService WSService;

	//http://localhost:8080/api/workshops

	@RequestMapping(value="/workshops", method=RequestMethod.GET)
	public List<WorkShops> readeWorkShops() {
```

```
		return WSService.getWorkShops();
	}

	//Get one WorkShop
	@RequestMapping(value="/workshop/{CodeW}", method=RequestMethod.GET)
	public WorkShops readeWorkShopById(@PathVariable(value = "CodeW")
Long codeW) {
		return WSService.getWorkShopByCodeW(codeW);
	}


}
```

WorkShopType.java

```
package carbook.webservice.CarBookAPI.model;


import javax.persistence.EmbeddedId;
import javax.persistence.Entity;
import javax.persistence.Table;

import org.springframework.stereotype.Component;


@Entity
@Table(name = "WorkShopType")

@Component
public class WorkShopType {

	@EmbeddedId
	WorkShopTypeId WST;


	public WorkShopTypeId getWST() {
		return WST;
	}

	public void setWST(WorkShopTypeId wST) {
		WST = wST;
	}




}
```

WorkShopTypeId.java

```
package carbook.webservice.CarBookAPI.model;

import java.io.Serializable;
import java.util.Objects;

import javax.persistence.Column;
import javax.persistence.Embeddable;

@Embeddable
```

```
public class WorkShopTypeId implements Serializable {


	public WorkShopTypeId() {
		super();
	}
	public WorkShopTypeId(Long codeW, String taskType) {
		super();
		this.codeW = codeW;
		this.taskType = taskType;
	}

	private static final long serialVersionUID = 1L;


	  @Column(name = "codeW", nullable = false)
	Long codeW;
	@Column(name = "taskType", nullable = false)
	String taskType;


	public Long getCodeW() {
		return codeW;
	}


	public void setCodeW(Long codeW) {
		this.codeW = codeW;
	}
	public String getTaskType() {
		return taskType;
	}
	public void setTaskType(String taskType) {
		this.taskType = taskType;
	}


	@Override
	public int hashCode() {
		return Objects.hash(codeW, taskType);
	}
	@Override
	public boolean equals(Object obj) {
		if (this == obj)
			return true;
		if (obj == null)
			return false;
		if (getClass() != obj.getClass())
			return false;
		WorkShopTypeId other = (WorkShopTypeId) obj;
		return Objects.equals(codeW, other.codeW) &&
Objects.equals(taskType, other.taskType);
	}

}
```

WorkShopTypeRepository.java

```
package carbook.webservice.CarBookAPI.repository;
```

```
import carbook.webservice.CarBookAPI.model.WorkShopType;
import carbook.webservice.CarBookAPI.model.WorkShopTypeId;

import java.util.List;

import org.springframework.data.repository.CrudRepository;
import org.springframework.stereotype.Repository;


@Repository
public interface WorkShopTypeRepository extends CrudRepository
<WorkShopType, WorkShopTypeId> {


      List<WorkShopType> findByWSTCodeW(Long codeW);
      List<WorkShopType> findByWSTTaskType(String taskType);



}
```

WorkShopTypeService.java

```
package carbook.webservice.CarBookAPI.service;

import carbook.webservice.CarBookAPI.model.WorkShopType;
import carbook.webservice.CarBookAPI.repository.WorkShopTypeRepository;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;



import java.util.List;
import java.util.stream.Collectors;
import java.util.stream.StreamSupport;



@Service
public class WorkShopTypeService {

      @Autowired
      WorkShopTypeRepository WSTrepository;

      public List<WorkShopType> getAll() {
        final Iterable<WorkShopType> iterable = WSTrepository.findAll();
        return StreamSupport.stream(iterable.spliterator(),
false).collect(Collectors.toList());
    }


      //find types by tasktype
    public List<WorkShopType> findByWSTTaskType(final String taskType) {
        return WSTrepository.findByWSTTaskType(taskType);
    }


  //find types by codeW
```

```
    public List<WorkShopType> findByWSTCodeW(final Long codeW) {
        return WSTrepository.findByWSTCodeW(codeW);
    }



}
```

WorkShopTypeController.java

```
package carbook.webservice.CarBookAPI.controller;

import carbook.webservice.CarBookAPI.model.WorkShopType;
import carbook.webservice.CarBookAPI.service.WorkShopTypeService;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.ResponseStatus;
import org.springframework.web.bind.annotation.RestController;


import org.springframework.http.HttpStatus;

import java.util.List;



@RestController
@RequestMapping("/api")
public class WorkShopTypeController {

      @Autowired
      WorkShopTypeService WSTservice;


      //http://localhost:8080/api/alltypes
      @GetMapping("/alltypes")
    @ResponseStatus(HttpStatus.OK)
    public List<WorkShopType> getWorkShopType() {
        return WSTservice.getAll();
    }



    //Get types by WorkShop

    @GetMapping("/wstype")
    @ResponseStatus(HttpStatus.OK)
    public List<WorkShopType> getWorkShopTypeByCodeW(@RequestParam(name =
"codeW") final Long codeW) {
        return WSTservice.findByWSTCodeW(codeW);
    }

    //Workshop code by tasktype
    @GetMapping("/tasktype")
    @ResponseStatus(HttpStatus.OK)
    public List<WorkShopType> getWorkShopTypeByTaskType(@RequestParam(name
= "taskType") final String taskType) {
```

```
        return WSTservice.findByWSTTaskType(taskType);
    }

}
```

AuthorizedWorkShops.java

```
package carbook.webservice.CarBookAPI.model;

import javax.persistence.Column;
import javax.persistence.EmbeddedId;
import javax.persistence.Entity;
import javax.persistence.Table;

import org.springframework.stereotype.Component;

@Entity
@Table(name = "AthorizedWorkShops")

@Component
public class AuthorizedWorkShops {

      @EmbeddedId
      AuthorizedWorkShopsId AWS;



    @Column(name="authorized")
       private int authorized;



      public int getAuthorized() {
            return authorized;
      }

      public void setAuthorized(int authorized) {
            this.authorized = authorized;
      }

      public Long getCodeW() {
            return AWS.codeW;
      }

      public void setCodeW(Long codeW) {
            this.AWS.codeW = codeW;
      }

      public Long getBrands() {
            return AWS.brands;
      }

      public void setBrands(Long brands) {
            this.AWS.brands = brands;
      }

}
```

AuthorizedWorkShopsId.java

```
package carbook.webservice.CarBookAPI.model;

import java.io.Serializable;
import java.util.Objects;
import javax.persistence.Column;
import javax.persistence.Embeddable;

@Embeddable
public class AuthorizedWorkShopsId implements Serializable {

	public AuthorizedWorkShopsId() {

	}
	public AuthorizedWorkShopsId(Long codeW, Long brands) {
		super();
		this.codeW = codeW;
		this.brands = brands;
	}
	private static final long serialVersionUID = 1L;

	@Column(name = "codeW", nullable = false)
    Long codeW;
    @Column(name = "brands", nullable = false)
    Long brands;


	public Long getCodeW() {
		return codeW;
	}
	public void setCodeW(Long codeW) {
		this.codeW = codeW;
	}
	public Long getBrand() {
		return brands;
	}
	public void setBrand(Long brands) {
		this.brands = brands;
	}

	@Override
	public int hashCode() {
		return Objects.hash(brands, codeW);
	}

	@Override
	public boolean equals(Object obj) {
		if (this == obj)
			return true;
		if (obj == null)
			return false;
		if (getClass() != obj.getClass())
			return false;
		AuthorizedWorkShopsId other = (AuthorizedWorkShopsId) obj;
		return Objects.equals(brands, other.brands) &&
Objects.equals(codeW, other.codeW);
	}
}
```

AuthorizedWorkShopsRepository.java

```
package carbook.webservice.CarBookAPI.repository;


import carbook.webservice.CarBookAPI.model.AuthorizedWorkShops;
import carbook.webservice.CarBookAPI.model.AuthorizedWorkShopsId;

import java.util.List;

import org.springframework.data.repository.CrudRepository;
import org.springframework.stereotype.Repository;



@Repository
public interface AuthorizedWorkShopsRepository extends CrudRepository
<AuthorizedWorkShops, AuthorizedWorkShopsId> {

	List<AuthorizedWorkShops> findByAWSCodeW(Long codeW);

	List<AuthorizedWorkShops> findByAWSBrands(Long brands);

}
```

AuthorizedWorkShopsService.java

```
package carbook.webservice.CarBookAPI.service;


import carbook.webservice.CarBookAPI.exception.EntityNotFound;
import carbook.webservice.CarBookAPI.model.AuthorizedWorkShops;
import carbook.webservice.CarBookAPI.model.AuthorizedWorkShopsId;
import
carbook.webservice.CarBookAPI.repository.AuthorizedWorkShopsRepository;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.util.List;
import java.util.stream.Collectors;
import java.util.stream.StreamSupport;



@Service
public class AuthorizedWorkShopsService {

	@Autowired
	AuthorizedWorkShopsRepository AWSrepository;



    public List<AuthorizedWorkShops> getAll() {
        final Iterable<AuthorizedWorkShops> iterable =
AWSrepository.findAll();
        return StreamSupport.stream(iterable.spliterator(),
false).collect(Collectors.toList());
    }
```

```
    //find by Composite Key
    public AuthorizedWorkShops findByCId(final Long codeW, final Long
brands)
            throws EntityNotFound {
        final AuthorizedWorkShopsId identity = new
AuthorizedWorkShopsId(codeW, brands);
        return
AWSrepository.findById(identity).orElseThrow(EntityNotFound::new);
    }


  //find for which brands is authorized by codeW
    public List<AuthorizedWorkShops> findByAWSCodeW(final Long codeW) {
        return AWSrepository.findByAWSCodeW(codeW);
    }

  //find for which workshops is authorized by brands
    public List<AuthorizedWorkShops> findByAWSBrands(final Long brands){
      return AWSrepository.findByAWSBrands(brands);
    }

}
```

AuthorizedWorkShopsController.java

```
package carbook.webservice.CarBookAPI.controller;


import carbook.webservice.CarBookAPI.exception.EntityNotFound;
import carbook.webservice.CarBookAPI.model.AuthorizedWorkShops;
import carbook.webservice.CarBookAPI.service.AuthorizedWorkShopsService;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.ResponseStatus;
import org.springframework.web.bind.annotation.RestController;
import org.springframework.http.HttpStatus;
import java.util.List;

@RestController
@RequestMapping("/api")
public class AuthorizedWorkShopsController {

      @Autowired
      AuthorizedWorkShopsService AWSservice;


      //URL - http://localhost:8080/api/all
      @GetMapping("/all")
    @ResponseStatus(HttpStatus.OK)
    public List<AuthorizedWorkShops> getAuthorizedWorkShops() {
        return AWSservice.getAll();
    }

    // composite key
    //URL - http://localhost:8080/api/authorized?codeW=1&brands=1
    @GetMapping("/authorizedbrand")
    @ResponseStatus(HttpStatus.OK)
```

```
    public AuthorizedWorkShops
getAuthorizedWorkShopsByCompositeKey(@RequestParam(name = "codeW") final
Long codeW,
                                         @RequestParam(name = "brands") final
Long brands)
            throws EntityNotFound {
        return AWSservice.findByCId(codeW, brands);
    }

//Get authorized brands by WorkShop
  //URL - http://localhost:8080/api/authorizedbrands?codeW=1

    @GetMapping("/authorizedbrands")
    @ResponseStatus(HttpStatus.OK)
    public List<AuthorizedWorkShops>
AuthorizedWorkShopsByCodeW(@RequestParam(name = "codeW") final Long codeW)
{
        return AWSservice.findByAWSCodeW(codeW);

    }

  //Get authorized WorkShops by brands
    //URL - http://localhost:8080/api/authorizedworkshops?brands=1
    @GetMapping("/authorizedworkshops")
    @ResponseStatus(HttpStatus.OK)
    public List<AuthorizedWorkShops>
AuthorizedWorkShopsByBrands(@RequestParam(name = "brands") final Long
brands) {
        return AWSservice.findByAWSBrands(brands);

    }

}
```

Πηγές και βιβλιογραφία

[1] Wikipedia. Λειτουργικό σύστημα (Διαδίκτυο). Αθήνα: Wikipedia; 2011 [ενημερώθηκε 2022 Ιανουάριος 18; Ανακτήθηκε 2021 Νοέμβριος 1]. Διαθέσιμο στον ιστότοπο: **https://el.wikipedia.org/wiki/Λειτουργικό_σύστημα**

[2] Coar. The Open Source Definition (Διαδίκτυο). Αθήνα: opensource; 2006 [ενημερώθηκε 2006 Αύγουστος 09; Ανακτήθηκε 2021 Νοέμβριος 1]. Διαθέσιμο στον ιστότοπο: **"The Open Source Definition"**.

[3] Wikipedia. Open source software (Διαδίκτυο). Αθήνα: Wikipedia; 2020 [ενημερώθηκε 2022 Αύγουστος 29; Ανακτήθηκε 2021 Νοέμβριος 1]. Διαθέσιμο στον ιστότοπο: https://en.wikipedia.org/wiki/Open_source - cite_ref-Diffingo_Solutions_Inc._2008_2-0 **"What is Open Source Software".**

[4] fandom. Operating system Android (Διαδίκτυο). Αθήνα: fandom; 2014 [ενημερώθηκε 2014 Δεκέμβριος 31; Ανακτήθηκε 2021 Νοέμβριος 1]. Διαθέσιμο στον ιστότοπο: **https://android.fandom.com/wiki/Android**

[5] Wikipedia. Android (Operating System) (Διαδίκτυο). Αθήνα: Wikipedia; 2020 [ενημερώθηκε 2022 Αύγουστος 29; Ανακτήθηκε 2021 Νοέμβριος 5]. Διαθέσιμο στον ιστότοπο: **https://en.wikipedia.org/wiki/Android_(operating_system)**

[6] Wikipedia. Open source (Διαδίκτυο). Αθήνα: Wikipedia; 2020 [ενημερώθηκε 2022 Αύγουστος 29; Ανακτήθηκε 2021 Νοέμβριος 5]. Διαθέσιμο στον ιστότοπο: **https://en.wikipedia.org/wiki/Open_source**

[7] Wikipedia. Android version history (Διαδίκτυο). Αθήνα: Wikipedia; 2012 [ενημερώθηκε 2022 Αύγουστος 31; Ανακτήθηκε 2021 Νοέμβριος 5]. Διαθέσιμο στον ιστότοπο: **https://en.wikipedia.org/wiki/Android_version_history#Android_1.0_(API_1)**

[8] Raphael JR . Android versions: A living history from 1.0 to 13 (Διαδίκτυο). Αμερική: Computerworld; 2015 [ενημερώθηκε 2022 Αύγουστος 23; Ανακτήθηκε 2021 Νοέμβριος 5]. Διαθέσιμο στον ιστότοπο: **https://www.computerworld.com/article/3235946/android-versions-a-living-history-from-1-0-to-today.html**

[9] Techopedia. Android Platform (Διαδίκτυο). Αθήνα: Techopedia; 2011 [ενημερώθηκε 2011 Αύγουστος 18; Ανακτήθηκε 2021 Νοέμβριος 5]. Διαθέσιμο στον ιστότοπο: **https://www.techopedia.com/definition/25102/android-eclair**

[10] Verge Staff. Android: A 10-year visual history (Διαδίκτυο). Αθήνα: The Verge; 2014 [ενημερώθηκε 2021 Δεκέμβριος 02; Ανακτήθηκε 2021 Νοέμβριος 5]. Διαθέσιμο στον ιστότοπο: **https://www.theverge.com/2011/12/7/2585779/android-10th-anniversary-google-history-pie-oreo-nougat-cupcake#QmHIj5**

[11] Kaiostech. Kaios operating system (Διαδίκτυο). Hong Kong: Kaiostech; 2017 [ενημερώθηκε 2018 Οκτώβριος 22; Ανακτήθηκε 2021 Νοέμβριος 5]. Διαθέσιμο στον ιστότοπο:
https://www.kaiostech.com

[12] Tutorialspoint. Android Architecture (Διαδίκτυο). Αθήνα: Tutorialspoint; 2011 [ενημερώθηκε 2020 Δεκέμβριος 29; Ανακτήθηκε 2021 Νοέμβριος 10]. Διαθέσιμο στον ιστότοπο:
https://www.tutorialspoint.com/android/android_architecture.htm

[13] Praveenruhil: Android Architecture (Διαδίκτυο). Noida: Geeks for geeks; 2010 [ενημερώθηκε 2022 Φεβρουάριος 26; Ανακτήθηκε 2021 Νοέμβριος 10]. Διαθέσιμο στον
https://www.geeksforgeeks.org/android-architecture/

[14] Google. Android Developers Application fundamentals (Διαδίκτυο). Αθήνα: Google; 2009 [ενημερώθηκε 2022 Σεπτέμβριος 2; Ανακτήθηκε 2021 Νοέμβριος 10]. Διαθέσιμο στον ιστότοπο: **https://developer.android.com/guide/components/fundamentals**

[15] Google. Android Developers Android Studio fundamentals (Διαδίκτυο). Αθήνα: Google; 2009 [ενημερώθηκε 2022 Σεπτέμβριος 2; Ανακτήθηκε 2021 Νοέμβριος 10]. Διαθέσιμο στον ιστότοπο: **https://developer.android.com/studio**

[16] Wikipedia. Android Software Development (Διαδίκτυο). Αθήνα: Wikipedia; 2008 [ενημερώθηκε 2022 Σεπτέμβριος 2; Ανακτήθηκε 2021 Νοέμβριος 10]. Διαθέσιμο στον ιστότοπο: **https://en.wikipedia.org/wiki/Android_software_development**

[17] Cardle J. P. Android App Development in Android Studio (Java + Android Edition for Beginners). 1η εκ. Manchester: Manchester Academic Publishers;2016

[18] Wikipedia. phpMyAdmin (Διαδίκτυο). Αθήνα: Wikipedia; 2014 [ενημερώθηκε 2022 Ιούλιος 27; Ανακτήθηκε 2021 Νοέμβριος 13]. Διαθέσιμο στον ιστότοπο: **https://en.wikipedia.org/wiki/PhpMyAdmin**

[19] Wikipedia. MySQL (Διαδίκτυο). Αθήνα: Wikipedia; 2012 [ενημερώθηκε 2022 Ιανουάριος 10; Ανακτήθηκε 2021 Νοέμβριος 13]. Διαθέσιμο στον ιστότοπο:
https://el.wikipedia.org/wiki/MySQL

[20] Wikipedia. SQLite (Διαδίκτυο). Αθήνα: Wikipedia; 2010 [ενημερώθηκε 2020 Ιούνιος 25; Ανακτήθηκε 2021 Νοέμβριος 13]. Διαθέσιμο στον ιστότοπο:
https://el.wikipedia.org/wiki/SQLite

[21] Wikipedia. Web Service Αθήνα: Wikipedia; 2017 [ενημερώθηκε 2022 Ιούλιος 20; Ανακτήθηκε 2021 Νοέμβριος 13]. Διαθέσιμο στον ιστότοπο:
https://en.wikipedia.org/wiki/Web_service

[22] Tutorialspoint.What are the Web Services Αθήνα: Tutorialspoint; 2017 [ενημερώθηκε 2022 Ιούλιος 29; Ανακτήθηκε 2021 Νοέμβριος 13]. Διαθέσιμο στον ιστότοπο:
https://www.tutorialspoint.com/webservices/what_are_web_services.htm

[23] Καρανικόλας Ν. Java για όλους. Αθήνα: Εκδόσεις Νέων Τεχνολογιών;2015

[24] Καρανικόλας Ν. Java για λίγους. Αθήνα: Εκδόσεις Νέων Τεχνολογιών;2018

[25] Έλληνας Ν. Ι. Εισαγωγή στον προγραμματισμό Android. Αθήνα: Εκδόσεις Τζιόλα;2014

[26] Harold E R. Προγραμματισμός δικτυακών εφαρμογών με Java. 4η εκ. Καβάλα: Εκδόσεις κλειδάριθμος;2016

[27] Αγιουτάντης Ζ. Γ., Μέρτικας Σ. Π. Ένας πρακτικός οδηγός για τη Συγγραφή Τεχνικών Κειμένων. Αθήνα: Εκδόσεις Ίων;2003

[28] Nissan. Nissan Micra εγχειρίδιο χρήσης Ιαπωνία: Nissan; 2017 [ενημερώθηκε 2021 Άυγουστος 02; Ανακτήθηκε 2021 Νοέμβριος 17]. Διαθέσιμο στον ιστότοπο: **https://www.nissan.gr/**

[29] Nissan. Nissan Leaf εγχειρίδιο χρήσης Ιαπωνία: Nissan; 2017 [ενημερώθηκε 2021 Αύγουστος 02; Ανακτήθηκε 2021 Νοέμβριος 17]. Διαθέσιμο στον ιστότοπο: **https://www.nissan.gr/**

[30] Fiat. Fiat Panda User Manual Ιταλία: Fiat; 2012 [ενημερώθηκε 2021 Μάρτιος 28; Ανακτήθηκε 2021 Νοέμβριος 17]. Διαθέσιμο στον ιστότοπο: **https://aftersales.fiat.com/eLumData/EN/00/319_PANDA/00_319_PANDA_603.99.249_EN_01_09.12_L_LG/00_319_PANDA_603.99.249_EN_01_09.12_L_LG.pdf**

[31] Fiat. Fiat Tipo User Manual Ιταλία: Fiat; 2015 [ενημερώθηκε 2020 Απρίλιος 06; Ανακτήθηκε 2021 Νοέμβριος 17]. Διαθέσιμο στον ιστότοπο: **https://www.fiat.ee/wp-content/uploads/2019/10/357_TIPO5P_EN.pdf**

[32] Smart. Smart ForTwo User Guide Γερμανία: 2014 [ενημερώθηκε 2022 Οκτώβριος 26; Ανακτήθηκε 2021 Νοέμβριος 17]. Διαθέσιμο στον ιστότοπο: **https://www.smart.mercedes-benz.com/gb/en/downloads#470**