Τεχνολογικό Εκπαιδευτικό Ίδρυμα Κρήτης



Σχολή Τεχνολογικών Εφαρμογών

Τμήμα Μηχανικών Πληροφορικής

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Τίτλος: Εφαρμογή Android για καθοδήγηση καθημερινών δραστηριοτήτων
(Android app for guiding daily activities)

Σπουδαστές

Παπαθεοδώρου Νικόλαος - Τσαμπίκος (ΑΜ: 4248)

Πέρρι Εντζίνα (ΑΜ: 4066)

Επιβλέπων εκπαιδευτικός: Παπαδάκης Νικόλαος

Ημερομηνία παρουσίασης: Φεβρουάριος, 2019

Πίνακας Περιεχομένων

Ευχαριστίες 3

Abstract 4

Σύνοψη 5

1. Εισαγωγή 6

1.1 Περίληψη 6

1.2 Κίνητρο για την Διεξαγωγή της Εργασίας – Στόχοι 6

1.3 Δομή Εργασίας 8

2. Μεθοδολογία Υλοποίησης 9

2.1 Μέθοδος Ανάλυσης & Ανάπτυξης 9

3. Σχέδιο Δράσης για την εκπόνηση της Πτυχιακής Εργασίας 15

3.1. Daily Guider στο Android Studio 15

3.1.1 XML 15

3.1.2 Java 17

3.1.3 SQLite 20

3.1.4 Adobe Photoshop 22

3.2 Σημαντικοί στόχοι για την ολοκλήρωση της Πτυχιακής Εργασίας 23

4. Κύριο μέρος Πτυχιακής Εργασίας 24

4.1 Ανάλυση Προβλήματος 24

4.1.2 Απαιτήσεις Συστήματος 24

4.2 Σχεδιασμός Υλοποίησης 25

4.3 Υλοποίηση 30

5. Αποτελέσματα 39

5.1 Συμπεράσματα 39

5.2 Μελλοντική Εργασία και Επεκτάσεις 40

Πίνακες…..42

Πίνακες εικόνων.....42

Βιβλιογραφία.....44

Παραρτήματα της Πτυχιακής Εργασίας.....45

Παράρτημα Α1 : DatabaseHelper.....45

Παράρτημα Α2 : Έλεγχος και εισαγωγή στοιχείων Χρήστη στη Βάση Δεδομένων....46

Παράρτημα Α3 : Μεταφορά από ένα Activity σε ένα άλλο.....46

Παράρτημα Α4 : Ολοκλήρωση Register Activity.....46

Παράρτημα Α5 : Έλεγχος Εγκυρότητας Στοιχείων Χρήστη.....47

Παράρτημα Α6 : Δημιουργία Χάρτη και Εύρεση τοποθεσίας Χρήστη.....48

Παράρτημα Α7 : Βήματα αναζήτησης πλησιέστερων τοποθεσιών.....49

Παράρτημα Α8 : Αναζήτηση κοντινότερου ανοικτού προορισμού.....50

Ευχαριστίες

Θα θέλαμε να ευχαριστήσουμε τον κύριο Παπαδάκη Νικόλαο για την καθοδήγηση που μας πρόσφερε καθ’ όλη τη διάρκεια εκπόνησης της πτυχιακής μας εργασίας.

Abstract

The application we developed provides guidance of services to any user who wants to do a daily activity. It will allow every user, if he/she has entered with his/her details, to enter or cancel an appointment as it will be given the map of the city with the distances and the hours of intense traffic or not, so that he/she can decide according to the current distance and if he/she can reach his/her destination quickly. The launching of activities will include cinemas, gas stations and supermarkets. Thus, the user will be able to do his day-to-day activities based on his/her schedule, identifying the closest places that interest him/her to go first and so save time.

Σύνοψη

Η εφαρμογή που αναπτύξαμε παρέχει υπηρεσίες καθοδήγησης σε οποιονδήποτε χρήστη που επιθυμεί να κάνει μια καθημερινή δραστηριότητα. Θα δίνει τη δυνατότητα σε κάθε χρήστη εφόσον έχει εγγραφεί με τα στοιχεία του, να εισάγει ή να ακυρώσει ένα ραντεβού του αφού θα υπάρχει ο χάρτης της πόλης με τις αποστάσεις και τις ώρες έντονης κυκλοφορίας ή όχι, ώστε να μπορεί να αποφασίσει ανάλογα με την εκάστοτε απόσταση που βρίσκεται και αν μπορεί να φτάσει γρήγορα στον προορισμό του. Η δρομολόγηση δραστηριοτήτων θα περιλαμβάνει τους κινηματογράφους, τα βενζινάδικα και τα σουπερμάρκετ. Έτσι, ο χρήστης θα μπορεί να κάνει τις καθημερινές του δραστηριότητες με βάση το πρόγραμμα του, εντοπίζοντας τα πιο κοντινά μέρη που τον ενδιαφέρουν να πάει πρώτα και άρα εξοικονομώντας χρόνο.

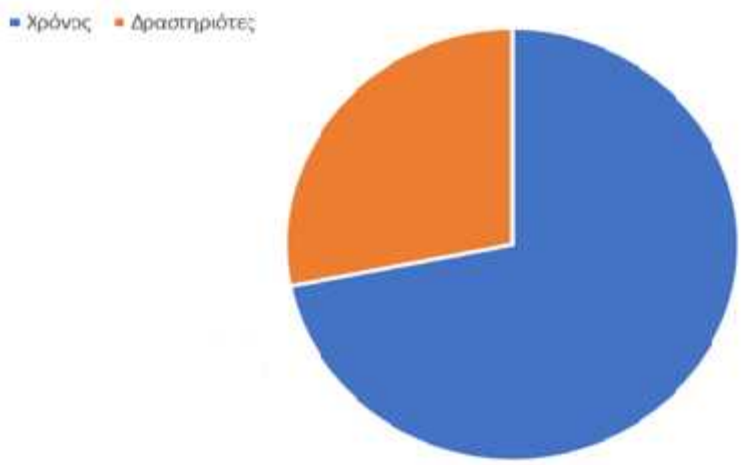
1. Εισαγωγή

1.1 Περίληψη

Η εφαρμογή που αναπτύξαμε παρέχει υπηρεσίες καθοδήγησης / δρομολόγησης σε οποιονδήποτε χρήστη (κυρίως ενός τουρίστα που δεν γνωρίζει το μέρος) που επιθυμεί να κάνει μια καθημερινή δραστηριότητα όπως είναι ο κινηματογράφος, το βενζινάδικο ή κάποιο σουπερμάρκετ. Θα υπάρχει ο χάρτης της πόλης με τις αποστάσεις και τις ώρες έντονης κυκλοφορίας ή όχι, ώστε να μπορεί να αποφασίσει ανάλογα με την απόσταση που βρίσκεται αν μπορεί να φτάσει γρήγορα στο μέρος που θέλει. Επομένως, ο χρήστης θα εντοπίζει τα πιο κοντινά μέρη που ενδιαφέρεται να πάει πρώτα με λιγότερο κόπο και χρόνο.

1.2 Κίνητρο για την Διεξαγωγή της Εργασίας – Στόχοι

Αρχικά, ο λόγος που επιλέξαμε το συγκεκριμένο θέμα είναι ότι οι καθημερινές δραστηριότητες των ανθρώπων καθώς ο χρόνος που τους παίρνει μέχρι να πάνε στο μέρος που θέλουν γρήγορα και χωρίς κόπο είναι σημαντικό κριτήριο που επηρεάζει και τις άλλες υποχρεώσεις τους. Ο χρόνος που ένα άτομο διαθέτει για να πάει σε ένα μέρος θεωρείται νεκρός και θα πρέπει να μην είναι υπερβολικός γιατί τότε υπάρχουν καθυστερήσεις στο υπόλοιπο του πρόγραμμα του καθώς και η τυχόν μη ολοκλήρωση κάποιας δραστηριότητας. Αυτό έχει ως αποτέλεσμα το άγχος στην καθημερινή του ζωή ως επακόλουθο τη δυσανασχέτηση του και τις συγκρούσεις με τον εαυτό του ή με τους συνανθρώπους του. Έτσι, η εφαρμογή που αναπτύξαμε έχει σκοπό τη διευκόλυνση των χρηστών (και κυρίως των τουριστών που δεν γνωρίζουν το μέρος) ώστε να υπολογίζουν μέσω του χάρτη που υπάρχει από το μέρος που βρίσκονται και να εντοπίζουν το πιο κοντινό μέρος σε αυτούς. Ταυτόχρονα, θα δίνεται ο χρόνος που χρειάζεται για τις μετακινήσεις του και θα χωρίζεται σε δύο χρονικά μέρη δηλαδή όταν υπάρχει ή όχι κυκλοφοριακό πρόβλημα και στην περίπτωση αυτή οι μετακινήσεις του θα είναι πιο αργές.



[ΕΙΚΟΝΑ 1 | ΔΙΑΓΡΑΜΜΑ ΑΠΑΙΤΗΣΗΣ ΧΡΟΝΟΥ - ΔΡΑΣΤΗΡΙΟΤΗΤΑΣ]

Έπειτα, ο λόγος που επιλέξαμε να αναπτύξουμε αυτή την εφαρμογή είναι ότι μας ενδιαφέρει ο χρήστης να οργανώνει τον χρόνο του και να βάζει σε σειρά προτεραιότητας τις καθημερινές του δραστηριότητες με βάση τις αποστάσεις που έχει να διανύσει μέχρι να φτάσει στο μέρος όπου επιθυμεί. Για το λόγο αυτό, αναπτύξαμε την εφαρμογή αυτή η οποία θα δίνει τη δυνατότητα σε κάθε χρήστη εφόσον έχει εγγραφεί με τα στοιχεία του, να εισάγει ή να ακυρώσει ένα ραντεβού του καθώς αν οι λοιπές υποχρεώσεις του επιτρέπουν την πραγματοποίηση του. Η δρομολόγηση θα περιλαμβάνει καθημερινές του δραστηριότητες όπως το να πάει σε έναν κινηματογράφο, σε ένα βενζινάδικο ή σε ένα σουπερμάρκετ. Αν δεν προλαβαίνει να πάει σε κάποιο από αυτά λόγω του ότι δεν προλαβαίνει λόγω κίνησης ή γιατί κλείνει το μέρος και θα βγαίνει αναγκαίο μήνυμα οπότε η εφαρμογή τον δρομολογεί να πάει στο πιο κοντινό του μέρος λαμβάνοντας υπόψιν και τις άλλες του υποχρεώσεις. Επομένως, ο χρήστης δεν θα ανησυχεί για το αν προλαβαίνει ή όχι να κάνει μια συγκεκριμένη δραστηριότητα αφού η εφαρμογή θα του λέει ακριβώς τι προλαβαίνει και τι όχι.

ΠΙΝΑΚΑΣ ΣΗΜΑΝΤΙΚΟΤΗΤΑΣ - ΕΠΕΙΓΟΝΤΟΣ		
	ΣΗΜΑΝΤΙΚΟ	ΜΗ ΣΗΜΑΝΤΙΚΟ
ΕΠΕΙΓΟΝ	Σοβαρά απρόβλεπτα γεγονότα	Πιεστικά θέματα της στιγμής
	Κρίσεις	Καθημερινές ενέργειες
	Τρία με προθεσμίες	
ΜΗ ΕΠΕΙΓΟΝ	Προγραμματισμός	Ενασχόληση με λεπτομέρειες
	Προετοιμασία	Ενασχόληση με ευχάριστες δραστηριότητες Αποφυγή δυσάρεστων
	Επαναπροσδιορισμός πορείας Ανάπτυξη δεξιοτήτων Αναψυχή	Ονειροπόληση

[ΕΙΚΟΝΑ 2 | ΠΙΝΑΚΑΣ ΣΗΜΑΝΤΙΚΟΤΗΤΑΣ - ΕΠΕΙΓΟΝΤΟΣ]

1.3 Δομή Εργασίας

Στην συνέχεια, ακολουθεί το κεφάλαιο 2 όπου θα αναλυθεί η μεθοδολογία υλοποίησης της εφαρμογής μας δηλαδή πως αναλύθηκε και αναπτύχθηκε. Μετά, ακολουθεί το κεφάλαιο 3, όπου θα αναπτυχθεί το σχέδιο δράσης που ακολουθήσαμε καθώς και οι σημαντικοί στόχοι για την ολοκλήρωση της εφαρμογής μας. Ύστερα, ακολουθεί το κεφάλαιο 4, όπου θα αναλυθεί το κύριο μέρος της εφαρμογής μας αναλύοντας το πρόβλημα μας, τις απαιτήσεις του συστήματος, το σχεδιασμό υλοποίησης και πως υλοποιήθηκε τελικά η εφαρμογή μας. Τέλος, ακολουθεί το κεφάλαιο 5, οπού θα αναλυθούν τα αποτελέσματα και συμπεράσματα της εργασίας και ακόμα πως θα μπορούσε να αναπτυχθεί μελλοντικά η εργασία μας με τρόπους βελτίωσης και με επεκτάσεις που θα μπορούσαν να γίνουν.

2. Μεθοδολογία Υλοποίησης

2.1 Μέθοδος Ανάλυσης & Ανάπτυξης

Για την ανάπτυξη της εφαρμογής μας χρησιμοποιήσαμε το εργαλείο Android Studio. Το Android Studio είναι ένα ολοκληρωμένο περιβάλλον ανάπτυξης (IDE) για το λειτουργικό σύστημα Android της Google, το οποίο βασίζεται στο λογισμικό IntelliJ IDEA της JetBrains και έχει σχεδιαστεί ειδικά για ανάπτυξη Android.

Είναι διαθέσιμο για λήψη σε λειτουργικά συστήματα που βασίζονται σε Windows, macOS και Linux. Αποτελεί αντικατάσταση του Eclipse Android Development Tools (ADT) ως το πρωταρχικό IDE για την ανάπτυξη εφαρμογών Android.

Το Android Studio ανακοινώθηκε στις 16 Μαΐου 2013 στο συνέδριο της Google I/O.

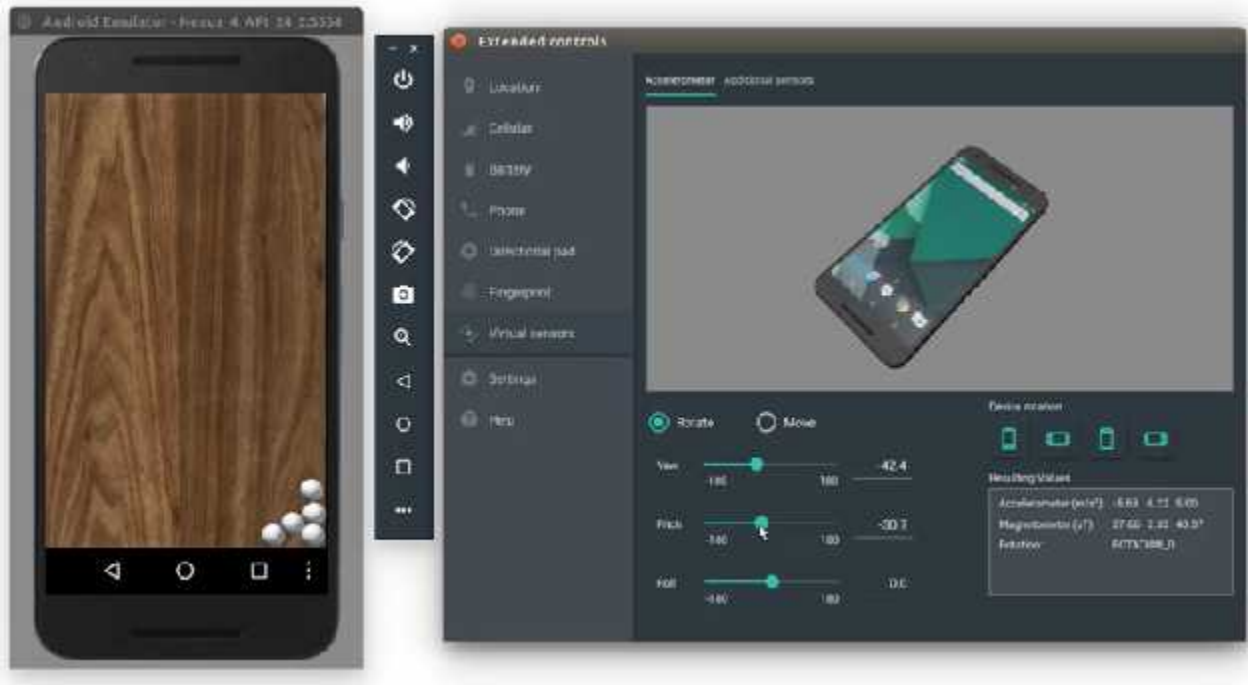


[ΕΙΚΟΝΑ 3 | ΛΟΓΟΤΥΠΟ ΠΡΟΓΡΑΜΜΑΤΟΣ ANDROID STUDIO]

Το Android Studio παρέχει βασικές λειτουργίες, όπως:

- Υποστήριξη οικοδόμησης με βάση το Gradle
- Συγκεκριμένα αντανακλαστικά για Android και γρήγορες επιδιορθώσεις
- Εργαλεία Lint για να πιάσουν την απόδοση, τη χρηστικότητα, τη συμβατότητα έκδοσης και άλλα προβλήματα
- Ενσωμάτωση ProGuard και δυνατότητες υπογραφής εφαρμογών
- Οδηγός με βάση το πρότυπο για τη δημιουργία κοινών σχεδίων και εξαρτημάτων Android
- Ένας πλούσιος επεξεργαστής διάταξης που επιτρέπει στους χρήστες να μεταφέρουν και να αποθέσουν στοιχεία UI, δυνατότητα προεπισκόπησης των σχημάτων σε διάφορες διαμορφώσεις οθόνης
- Υποστήριξη για την ανάπτυξη εφαρμογών Android Wear

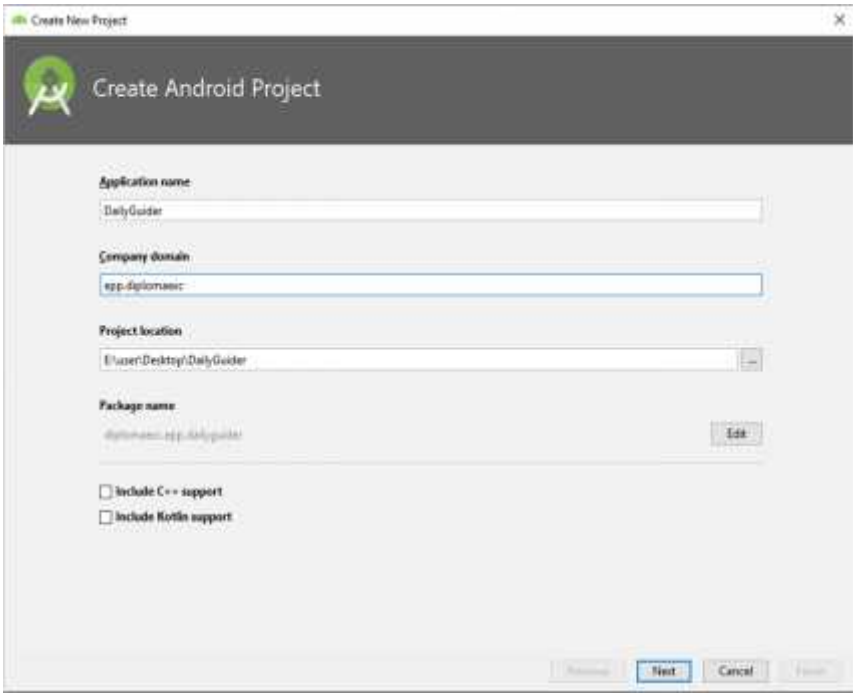
- Ενσωματωμένη υποστήριξη για την πλατφόρμα Google Cloud που επιτρέπει την ενσωμάτωση με τα μηνύματα Cloud Messages και το Google App Engine
- Εικονική συσκευή Android (Emulator) για την εκτέλεση και τον εντοπισμό σφαλμάτων σε εφαρμογές στο Android Studio, όπως φαίνεται στην εικόνα 4.



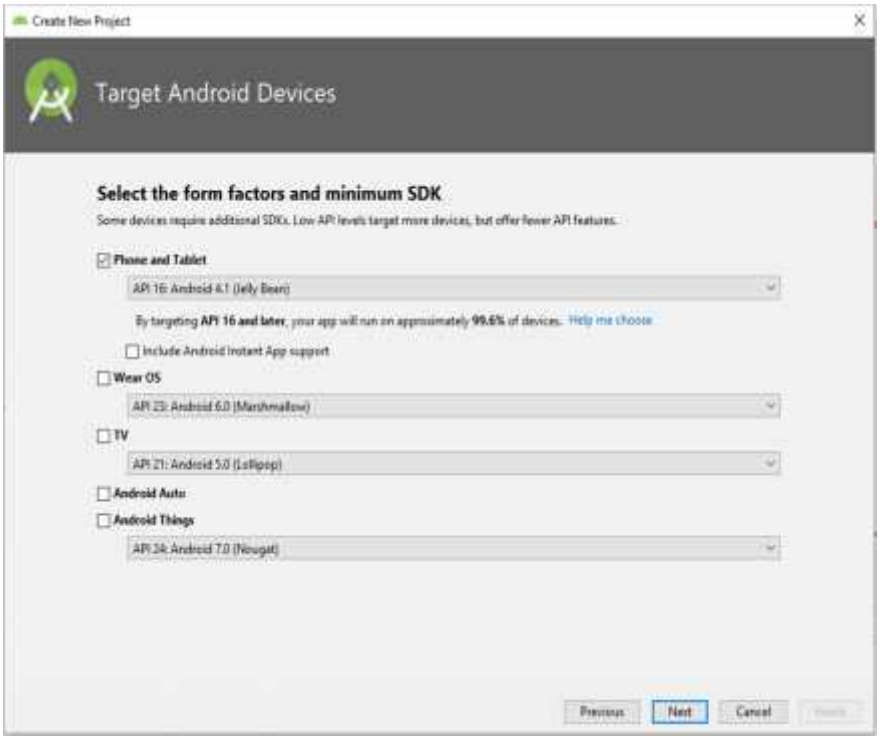
[ΕΙΚΟΝΑ 4 | ΕΙΚΟΝΙΚΗ ΣΥΣΚΕΥΗ ANDROID (EMULATOR)]

Το Android Studio υποστηρίζει τις γλώσσες προγραμματισμού του IntelliJ και CLion όπως την Java και τη C++.

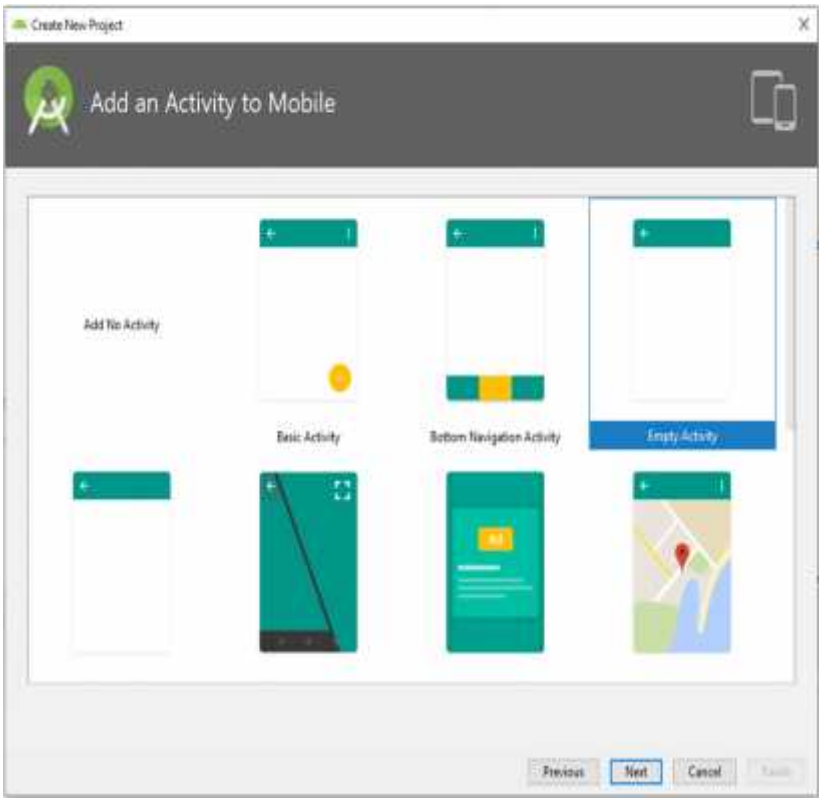
Αρχικά, μετά την εγκατάσταση του εργαλείου Android Studio δημιουργήσαμε μια νέα εφαρμογή Android, το οποίο θα μας φέρει στο παράθυρο που φαίνεται παρακάτω και θα πρέπει να διαλέξουμε τις επιλογές, που φαίνονται στην εικόνα 5, 6, 7, 8.



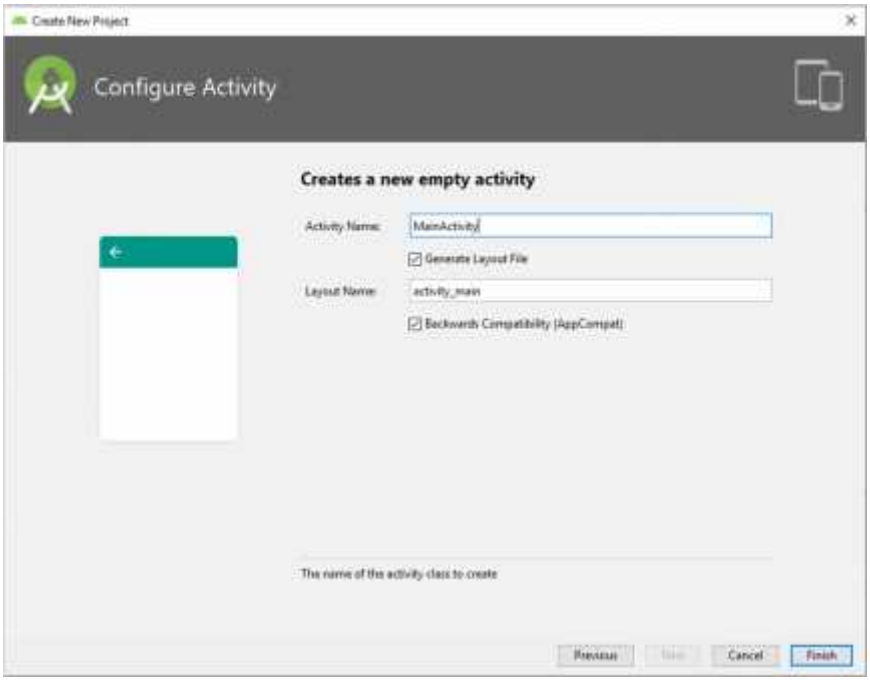
[ΕΙΚΟΝΑ 5 | ΔΕΙΓΜΑ ΔΗΜΙΟΥΡΓΙΑΣ ΕΝΩΣ ΝΕΟΥ ΠΡΟΤΖΕΚΤ (1)]



[ΕΙΚΟΝΑ 6 | ΔΕΙΓΜΑ ΔΗΜΙΟΥΡΓΙΑΣ ΕΝΩΣ ΝΕΟΥ ΠΡΟΤΖΕΚΤ (2)]



[ΕΙΚΟΝΑ 7 | ΔΕΙΓΜΑ ΔΗΜΙΟΥΡΓΙΑΣ ΕΝΩΣ ΝΕΟΥ ΠΡΟΤΖΕΚΤ (3)]



[ΕΙΚΟΝΑ 8 | ΔΕΙΓΜΑ ΔΗΜΙΟΥΡΓΙΑΣ ΕΝΩΣ ΝΕΟΥ ΠΡΟΤΖΕΚΤ (4)]

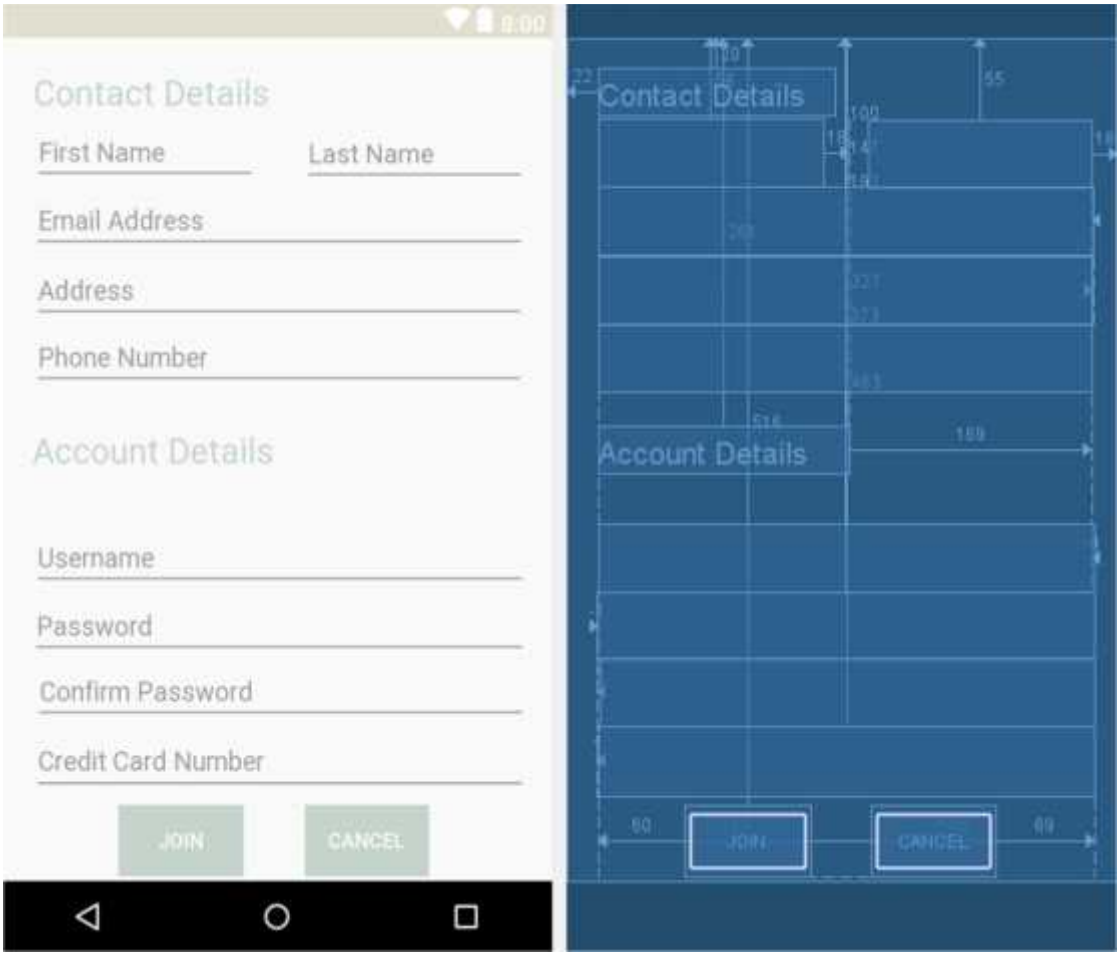
Στην συνέχεια, θα γράψουμε το XML κώδικα επιλέγοντας πρώτα το όνομα της εφαρμογής μας, μετά ξεκινάμε να συγγράφουμε στο XML κώδικα με βάση την εμφάνιση της εφαρμογής μας που έχουμε στο μυαλό μας και κάνοντας τα αναπαράσταση στον Emulator του Android Studio, όπως φαίνεται στην εικόνα 9.



[ΕΙΚΟΝΑ 9 | ΤΟ ΚΕΝΤΡΙΚΟ ΜΕΝΟΥ ΤΗΣ ΕΦΑΡΜΟΓΗΣ ΜΑΣ ΑΠΟ ΤΟ VISUAL LAYOUT EDITOR]

Πλέον, είμαστε έτοιμοι για τη συγγραφή του κώδικα μας σε γλώσσα Java.

Ταυτόχρονα, θα χρησιμοποιήσουμε την SQLite η οποία είναι μια βιβλιοθήκη γλώσσας C που υλοποιεί μια μικρή, γρήγορη, αυτόνομη, υψηλής αξιοπιστίας, πλήρους μηχανής, βάση δεδομένων SQL. Στόχος είναι να αποθηκεύσουμε το όνομα και το επώνυμο του χρήστη, τη διεύθυνση, το τηλέφωνο του, τον αριθμό της πιστωτικής και ένα μοναδικό κωδικό τον οποίο τον πληκτρολογεί ο χρήστης και για επιβεβαίωση τον ξανά πληκτρολογεί για να επαληθεύσει εάν είναι ίδιος με τον αυτόν που πληκτρολόγησε αρχικά. Επίσης, ο χρήστης θα έχει ένα username και ένα password, όπως φαίνεται στην εικόνα 10.



[ΕΙΚΟΝΑ 10 | ΤΟ ΜΕΝΟΥ ΕΓΓΡΑΦΗΣ ΤΗΣ ΕΦΑΡΜΟΓΗΣ ΜΑΣ ΑΠΟ ΤΟ VISUAL LAYOUT EDITOR]

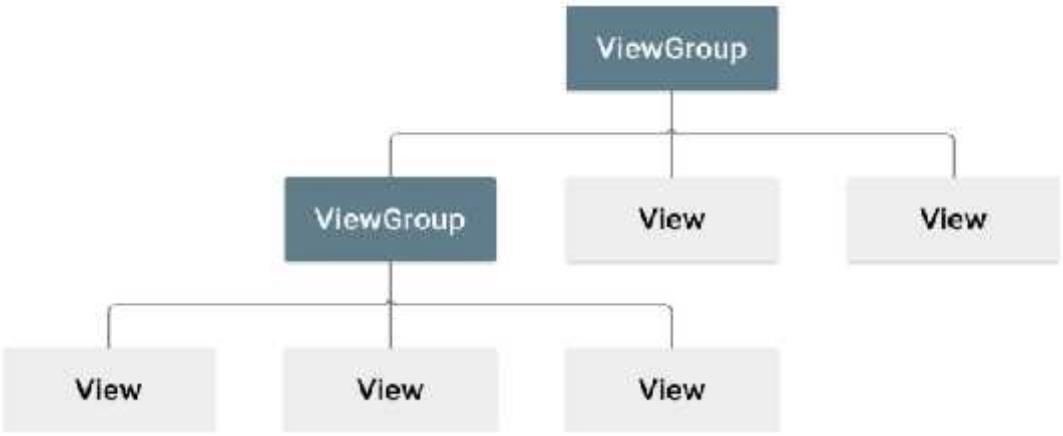
3. Σχέδιο Δράσης για την εκπόνηση της Πτυχιακής Εργασίας

3.1. Daily Guider στο Android Studio

3.1.1 XML

Το XML αντιπροσωπεύει το eXtensible Markup Language δηλαδή την εκτατή γλώσσα σήμανσης. Η XML είναι μια αυτοπεριγραφική γλώσσα σήμανσης παρόμοια με το HTML και σχεδιάστηκε για την αποθήκευση και τη μεταφορά δεδομένων.

Μια διάταξη ορίζει τη δομή για μια διεπαφή χρήστη στην εφαρμογή. Όλα τα στοιχεία στη διάταξη κατασκευάζονται χρησιμοποιώντας μια ιεραρχία αντικειμένων View και ViewGroup. Ένα View συνήθως αντλεί κάτι που ο χρήστης μπορεί να δει και να αλληλεπιδράσει. Ενώ ένα ViewGroup είναι ένα αόρατο δοχείο που καθορίζει τη δομή της διάταξης για την προβολή και άλλων αντικειμένων της ViewGroup.



[ΕΙΚΟΝΑ 11 | VIEWGROUP]

Τα View αντικείμενα ονομάζονται συνήθως "γραφικά στοιχεία" και μπορούν να είναι μια από τις πολλές υποκατηγορίες, όπως είναι το Button ή το TextView. Τα αντικείμενα της ViewGroup ονομάζονται συνήθως "διατάξεις" και μπορεί να είναι ένας από τους πολλούς τύπους που παρέχουν διαφορετική δομή διάταξης, όπως το LinearLayout ή το ConstraintLayout.

Μια διάταξη μπορεί να δηλωθεί με δύο τρόπους:

- Δήλωση στοιχείων διεπαφής χρήστη (UI) σε XML. Το Android παρέχει ένα απλό λεξιλόγιο XML που αντιστοιχεί στις κλάσεις View και τις υποκλάσεις, όπως αυτές για widgets και layouts. Επίσης, μπορεί να χρησιμοποιηθεί Layout Editor του Android

Studio για να δημιουργηθεί η διάταξη XML χρησιμοποιώντας μια διεπαφή drag and drop.

- Στοιχεία layout παραστατικών κατά το χρόνο εκτέλεσης. Η εφαρμογή μπορεί να δημιουργήσει προγραμματιστικά αντικείμενα View και ViewGroup.

Η δήλωση της διεπαφή χρήστη (UI) σε XML επιτρέπει να διαχωριστεί η παρουσίαση της εφαρμογής από τον κώδικα που ελέγχει τη συμπεριφορά της. Η χρήση αρχείων XML διευκολύνει, ακόμη, την παροχή διαφορετικών σχεδιαγραμμάτων για διαφορετικά μεγέθη και προσανατολισμούς οθόνης.

Αρχικά, γράφουμε το XML κώδικα επιλέγοντας πρώτα το όνομα της εφαρμογής μας ως «Daily Guider». Στη συνέχεια, ξεκινάμε να συγγράφουμε στο XML κώδικα με βάση την εμφάνιση της εφαρμογής μας που έχουμε στο μυαλό μας, που αποθηκεύεται σε αρχείο (.xml).

Για παράδειγμα, η δομή ενός XML κώδικα είναι ως εξής:



[ΕΙΚΟΝΑ 12 | ΠΑΡΑΔΕΙΓΜΑ ΚΩΔΙΚΑ XML]

Ταυτόχρονα, κάνουμε την αναπαράσταση στον Emulator του Android Studio, όπως φαίνεται στην εικόνα 13,14,15.



[ΕΙΚΟΝΑ 13 | EMULATOR MAIN]



[ΕΙΚΟΝΑ 14 | EMULATOR REGISTER]



[ΕΙΚΟΝΑ 15 | EMULATOR MAP]

3.1.2 Java

Η Java είναι μία αντικειμενοστραφής γλώσσα προγραμματισμού, η οποία βασίζεται στις κλάσεις και στα αντικείμενα. Έχει πολλά πλεονεκτήματα έτσι όπως είναι σχεδιασμένη και ειδικά στο Android Studio όταν αναπτύσσει κάποιος μια εφαρμογή. Όταν ανοίξει νέο παράθυρο με όνομα «Create New Class», μπορεί κάποιος εύκολα να δημιουργήσει τις ακόλουθες νέες κλάσεις και τύπους:

- Java κλάσεις
- Καταμέτρηση και singleton κλάσεις
- Τύποι διεπαφής και σχολιασμού

Αφού συμπληρωθούν τα πεδία διαλόγου στο «Create New Class» και γίνει κλικ στο κουμπί OK, το Android Studio δημιουργεί ένα αρχείο (.java) που περιέχει κώδικα σκελετού, ο οποίος

περιλαμβάνει μια δήλωση πακέτου, όλες τις απαραίτητες εισαγωγές, μια κεφαλίδα και μια δήλωση κλάσης ή τύπου. Στη συνέχεια, μπορεί να προστεθεί κωδικός σε αυτό το αρχείο.

Τα πρότυπα αρχείων καθορίζουν τον τρόπο με τον οποίο το Android Studio παράγει τον κώδικα σκελετού. Μπορεί να χρησιμοποιηθούν τα πρότυπα αρχείων που παρέχονται στο Android Studio όπως είναι ή να τα προσαρμοστούν ώστε να ταιριάζουν με την αναπτυξιακή διαδικασία της εκάστοτε εφαρμογής.

Παρακάτω, θα παρουσιάσουμε βασικά μέρη της Java έτσι ώστε να την κατανοήσουμε και να την αναπτύξουμε σωστά.

Αρχικά, θα αναπτύξουμε τις βασικές έννοιες της Java:

- Οι **κλάσεις** και ο τρόπος δήλωσης των μεταβλητών της κλάσης (Class Variables) και μεθόδων.

```
1   package javatutorial;
2
3   public class Car{
4
5       String name;
6       int gear;
7       double speed;
8
9       void Accelerate(){
10      }
11
12      void SlowDown(){
13      }
14
15  }
```

[ΕΙΚΟΝΑ 16 | JAVATUTORIAL ΚΛΑΣΕΙΣ]

- Ο **constructor** είναι μια μέθοδος που χρησιμοποιείται για την αρχικοποίηση του αντικειμένου καθώς και για την αρχικοποίηση άλλων παραμέτρων μέσα στο αντικείμενο. Γενικά, ένα αντικείμενο μπορεί να έχει περισσότερους από έναν constructor, με διαφορετική παραμετροποίηση ο καθένας από αυτούς.

```
1   package javatutorial;
2
3   public class Car{
4
5       string name;
6       int gear;
7       double speed;
8
9       public void car (String Cname){
10          name=Cname;
11      }
12
13  }
```

[ΕΙΚΟΝΑ 17 | JAVATUTORIAL CONSTRUCTOR]

- Η **κληρονομικότητα** δηλαδή όταν μια κλάση έχει ήδη χαρακτηριστικά μιας άλλης κλάσης. Επίσης, η νέα κλάση μπορεί να διαχειριστεί μεταβλητές κλάσης (Class Variables) χωρίς να χρειάζεται να δηλωθούν εφόσον κληρονομούνται από την άλλη κλάση.

```
1   package javatutorial;
2
3   public class Opel extends Car{
4
5       public void Opel(){
6           this.name="Astra";
7       }
8
9   }
```

[ΕΙΚΟΝΑ 18 | JAVATUTORIAL ΚΛΗΡΟΝΟΜΙΚΟΤΗΤΑ]

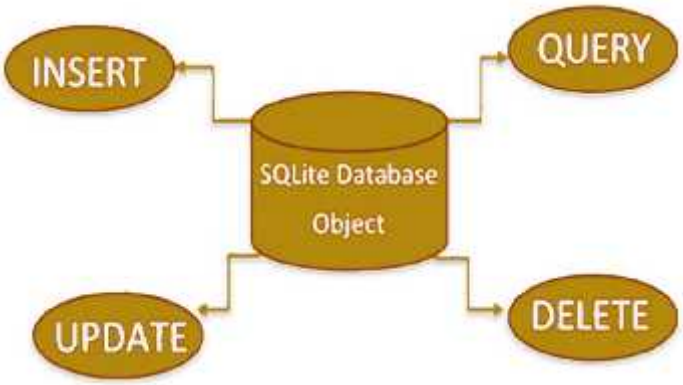
- **Απόκρυψη πληροφορίας** (Data Hiding) δηλαδή χρησιμοποιώντας την λέξη private δηλώνουμε μια μεταβλητή ως ιδιωτική που δίνει τη δυνατότητα στο αντικείμενο να προστατεύει τις πληροφορίες από το να αλλάζουν από μεθόδους εκτός του ίδιου του αντικειμένου.

```
1   package javatutorial;
2
3   public class Car{
4
5       private String name;
6       private int gear;
7       private double speed;
8
9       public void car(String Cname){
10          name=Cname;
11      }
12
13  }
```

[ΕΙΚΟΝΑ 19 | ΑΠΟΚΡΥΨΗ ΠΛΗΡΟΦΟΡΙΑΣ]

3.1.3 SQLite

SQLite είναι ένα σύστημα διαχείρισης σχεσιακής βάσης δεδομένων (RDBMS). Είναι δηλαδή μία δομή βάσης ερωτημάτων βάσεων δεδομένων, ανοιχτού κώδικα, ελαφρύ όγκου, μηδενικής πρόσβασης σε δίκτυο και αυτόνομης βάσης δεδομένων. Υποστηρίζει ενσωματωμένες λειτουργίες σχεσιακής βάσης δεδομένων.



[ΕΙΚΟΝΑ 20 | SQLITE DATABASE OBJECT]

Κάθε φορά που μια εφαρμογή χρειάζεται να αποθηκεύει μεγάλο όγκο δεδομένων, τότε η χρήση του SQLite είναι προτιμότερη από ό, τι άλλο σύστημα αποθετηρίων όπως το SharedPreferences ή η αποθήκευση δεδομένων σε αρχεία.

Το Android Studio έχει ενσωματώσει την υλοποίηση βάσεων δεδομένων SQLite. Διατίθεται τοπικά στη συσκευή (κινητό και tablet) και περιέχει δεδομένα σε μορφή κειμένου. Φέρνει δεδομένα ελαφρού βάρους και είναι κατάλληλο για πολλές γλώσσες. Επομένως, δεν απαιτεί καμία διαδικασία διαχείρισης ή ρύθμισης της βάσης δεδομένων.

Android SQLite Java κλάσεις

Cursor είναι μια κλάση παρέχει πρόσβαση στα αποτελέσματα ενός ερωτήματος βάσης δεδομένων.
Οι μέθοδοι περιλαμβάνουν:
close (): απελευθερώνει όλους τους πόρους που χρησιμοποιούνται από το δρομέα και τους κλείνει.
getCount (): επιστρέφει τον αριθμό των γραμμών που περιέχονται στο σύνολο αποτελεσμάτων.
moveToFirst (): μεταβαίνει στην πρώτη σειρά στο σύνολο αποτελεσμάτων.
moveToLast (): μεταβαίνει στην τελευταία σειρά στο σύνολο αποτελεσμάτων.
moveToNext (): μεταβαίνει στην επόμενη σειρά στο σύνολο αποτελεσμάτων.
move (): μετακινείται με μια καθορισμένη μετατόπιση από την τρέχουσα θέση στο σύνολο αποτελεσμάτων.
get <type> (): (όπως getInt (), getDouble (), κ.λπ.) επιστρέφει την τιμή του καθορισμένου <type> που περιέχεται στον καθορισμένο δείκτη στήλης της σειράς στην τρέχουσα θέση δρομέα.

Το SQLiteDatabase παρέχει την κύρια διεπαφή μεταξύ του κώδικα εφαρμογής και της υποκείμενης βάσης δεδομένων SQLite.
Οι μέθοδοι περιλαμβάνουν:
insert (): εισάγει μια νέα σειρά σε έναν πίνακα βάσης δεδομένων.
delete (): διαγράφει γραμμές από έναν πίνακα βάσης δεδομένων
query (): εκτελεί ένα συγκεκριμένο ερώτημα βάσης δεδομένων και επιστρέφει τα αντίστοιχα αποτελέσματα μέσω ενός αντικειμένου Cursor.
execSQL (): εκτελεί μια ενιαία δήλωση SQL που δεν επιστρέφει δεδομένα αποτελεσμάτων.
rawQuery (): εκτελεί μια εντολή ερώτησης SQL και επιστρέφει αντίστοιχα αποτελέσματα με τη μορφή ενός αντικειμένου Cursor.

Το SQLiteOpenHelper έχει σχεδιαστεί για να διευκολύνει τη δημιουργία και την ενημέρωση βάσεων δεδομένων.
Οι μέθοδοι περιλαμβάνουν:
onCreate (): καλείται όταν δημιουργείται η βάση δεδομένων για πρώτη φορά.
onUpgrade (): ονομάζεται σε περίπτωση που ο κώδικας εφαρμογής περιέχει πιο πρόσφατη αναφορά έκδοσης βάσης δεδομένων.
onOpen (): καλείται όταν ανοίγει η βάση δεδομένων.
getWritableDatabase (): ανοίγει ή δημιουργεί βάση δεδομένων για ανάγνωση και γραφή.
getReadableDatabase (): δημιουργεί ή ανοίγει μια βάση δεδομένων για ανάγνωση μόνο.
close (): κλείνει τη βάση δεδομένων.

Το ContentValues επιτρέπει την αναγραφή ζευγών κλειδιού/τιμής που αποτελείται από ταυτοποιητές στήλης πίνακα και τις τιμές που πρέπει να αποθηκεύονται σε κάθε στήλη.
Οι μέθοδοι περιλαμβάνουν:
put (): προσθέτει μια τιμή στο σύνολο.

3.1.4 Adobe Photoshop

Το Adobe Photoshop είναι ένας raster graphics editor που αναπτύχθηκε και δημοσιεύθηκε από το Adobe Systems για το Mac OS και τα Windows.

Είναι το κυρίαρχο λογισμικό επεξεργασίας και χειρισμού φωτογραφιών στην αγορά. Οι χρήσεις του κυμαίνονται από την πλήρη εμφάνιση μεγάλων παρτίδων φωτογραφιών στη δημιουργία περίπλοκων ψηφιακών ζωγραφιών και σχεδίων που μιμούνται αυτά που γίνονται με το χέρι.



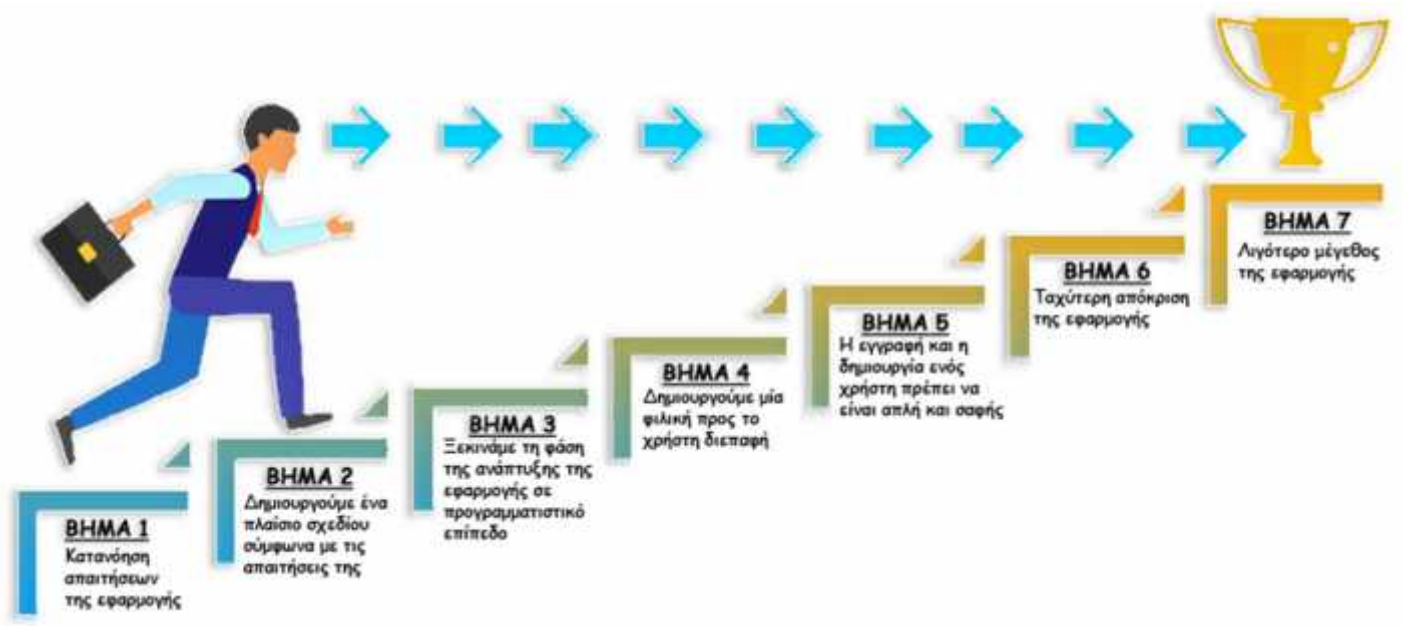
[ΕΙΚΟΝΑ 21 | ADOBE PHOTOSHOP LOGO]

Στην εφαρμογή μας χρησιμοποιήσαμε το Adobe Photoshop για την δημιουργία και επεξεργασία των εικονιδίων.

3.2 Σημαντικοί στόχοι για την ολοκλήρωση της Πτυχιακής Εργασίας

Για να επιτύχουμε τους στόχους για την ολοκλήρωση της εφαρμογής μας πρέπει να ακολουθήσουμε τα εξής βήματα:

1. Κατανόηση των απαιτήσεων της εφαρμογής μας
2. Δημιουργούμε ένα πλαίσιο σχεδίου σύμφωνα με τις απαιτήσεις της
3. Ξεκινάμε τη φάση της ανάπτυξης της εφαρμογής σε προγραμματιστικό επίπεδο
4. Δημιουργούμε μία φιλική προς τον χρήστη διεπαφή
5. Η εγγραφή και η δημιουργία ενός χρήστη πρέπει να είναι απλή και σαφής
6. Ταχύτερη απόκριση της εφαρμογής
7. Λιγότερο μέγεθος της εφαρμογής



[ΕΙΚΟΝΑ 22 | ΣΤΟΧΟΙ ΟΛΟΚΛΗΡΩΣΗΣ ΕΦΑΡΜΟΓΗΣ]

4. Κύριο μέρος Πτυχιακής Εργασίας

4.1. Ανάλυση Προβλήματος

Προκειμένου να ξεκινήσουμε ένα Activity στο Android Studio, θα πρέπει πρώτα να φτιάξουμε τη διεπαφή μας μέσω του Visual layout Editor που παρέχεται από το Android Studio. Στη συνέχεια, θα αναπτύξουμε τον κώδικα Java σε συνδυασμό με SQLite για την αποθήκευση και ανάκτηση δεδομένων. Ταυτόχρονα, για την εύρεση τοποθεσιών χρησιμοποιήσαμε κάποια από τα APIs της Google και πιο συγκεκριμένα το Directions API, το Places API. Επιπλέον, χρησιμοποιήθηκε το Map SDK for Android. Παράλληλα, για την διεπαφή μας δημιουργήθηκαν κάποια εικονίδια για την εφαρμογή μας από εμάς μέσω του προγράμματος και σχεδιασμού του Photoshop. Παρακάτω θα εμβαθύνουμε σε κάθε ένα από αυτά αναλύοντας τη διαδικασία που ακολουθήσαμε για την ανάπτυξη της εφαρμογής μας Daily Guider.

Αρχικά, αναλύσαμε το πρόβλημα μας σε μικρότερα υπό προβλήματα ώστε να γίνει εύκολή και ποιο γρήγορη η υλοποίηση του. Το πρώτο υπό πρόβλημα μας ήταν να κατανοήσουμε τις ανάγκες και τις χρησιμότητες της εφαρμογής μας προς τον χρήστη. Στην συνέχεια, μελετήσαμε το σχεδιασμό της διεπαφής μας, ώστε να είναι όσο γίνεται ποιο κατανοητή και εύχρηστή προς τον χρήστη. Μια κατανοητή και εύχρηστη διεπαφή είναι αναγκαία για την εφαρμογή μας, διότι είναι μια εφαρμογή άνευ ορίου ηλικίας. Αφού σκεφτήκαμε την μορφολογία των «παραθύρων» και γενικότερα της διεπαφής μας, το επόμενο βήμα ήταν να σκεφτούμε το πως θα είναι το «γραφικό» κομμάτι της εφαρμογής μας με αυτό να συνεπάγει όλα τα εικονίδια που θα χρησιμοποιηθούν μέσα σε αυτή. Μετά από την επιτυχημένη υλοποίηση των παραπάνω υπό προβλημάτων, έπρεπε να αντιμετωπίσουμε το πρόβλημα αφού πρώτα το σχεδιάσουμε και έπειτα το υλοποιήσουμε.

4.1.2 Απαιτήσεις Συστήματος

Για να υλοποιηθεί η εφαρμογή μας χρειαστήκαμε έναν εξοπλισμό ο οποίος εκπλήρωνέ την απαίτηση όλων των αναγκαίων χρησιμοποιημένων προγραμμάτων, των οποίων η βοήθεια ήταν απαραίτητη για την ολοκληρωμένη και σωστή υλοποίηση της εφαρμογής μας. Τα προγράμματα που χρησιμοποιήσαμε ήταν το Android Studio, το DB Browser for SQLite και το Adobe Photoshop. Το ποιο απαιτητικό από τα παραπάνω προγράμματα ήταν το Android Studio, που αυτό σημαίνει ότι αν εκπληρώσουμε τις απαιτήσεις του Compiler μας θα έχουμε εκπληρώσει και τις απαιτήσεις των άλλων προγραμμάτων. Οι απαιτήσεις του Android Studio ήταν ένα λογισμικό της Microsoft νέας γενιάς (π.χ. Windows 7/8/10), 8GB RAM με επιπλέον άλλο 1GB για τον

Emulator του Android, 4GB ελεύθερου χώρου στον σκληρό δίσκο (500MB για το IDE και 1.5GB για το Android SDK και το emulator system image) και τέλος μια ανάλυση μεγαλύτερη η ίση των 1280 x 800 pixel. Στην συνέχεια χρειαστήκαμε ένα Smartphone για να δοκιμάσουμε την τελική έκδοση της εφαρμογής μας. Τα απαραίτητα χαρακτηριστικά του κινητού ήταν ένα λογισμικό Android έκδοσης Jelly Bean η μεγαλύτερη, 2GB RAM, GPS και υποστήριξη των υπηρεσιών της Google.

4.2 Σχεδιασμός Προβλήματος

Αρχικά, για τον σχεδιασμό του προβλήματος σκεφτήκαμε και επεξεργαστήκαμε στο Adobe Photoshop τα εικονίδια που θα χρησιμοποιήσουμε στην εφαρμογή μας και χωρίζονται στα παρακάτω:

- Τα εικονίδια για το λογότυπο της εφαρμογής μας



[ΕΙΚΟΝΑ 23 | ΛΟΓΟΤΥΠΑ ΤΗΣ ΕΦΑΡΜΟΓΗΣ]

- Το εικονίδιο για τα πρατήρια αυτοκινήτων (ή βενζινάδικα)



[ΕΙΚΟΝΑ 24 | ΕΙΚΟΝΙΔΙΟ ΒΕΝΖΙΝΑΔΙΚΟΥ]

- Το εικονίδιο για τα σουπερμάρκετ



[ΕΙΚΟΝΑ 25 | ΕΙΚΟΝΙΔΙΟ SUPERMARKET]

- Το εικονίδιο για τον κινηματογράφο



[ΕΙΚΟΝΑ 26 | ΕΙΚΟΝΙΔΙΟ ΚΙΝΗΜΑΤΟΓΡΑΦΟΥ]

- Το εικονίδιο το οποίο μας οδηγεί στο Μενού των ραντεβού



[ΕΙΚΟΝΑ 27 | ΕΙΚΟΝΙΔΙΟ ΜΕΝΟΥ ΡΑΝΤΕΒΟΥ]

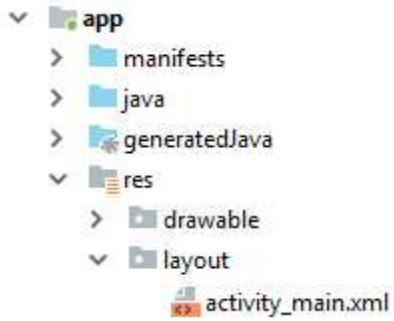
- Το εικονίδιο για την Αναζήτηση τοποθεσίας



[ΕΙΚΟΝΑ 28 | ΕΙΚΟΝΙΔΙΟ ΑΝΑΖΗΤΗΣΗΣ ΤΟΠΟΘΕΣΙΑΣ]

Στην συνέχεια, ασχοληθήκαμε με τον σχεδιασμό της διεπαφής της εφαρμογής μας και ακολουθήσαμε τα εξής βήματα:

1) Μετά την δημιουργία του πρότζεκτ μας και συγκεκριμένα της πρώτης κλάσης μας με όνομα Main Activity όπως είδαμε προηγουμένως στις εικόνες 5, 6, 7, 8 πήγαμε στον φάκελο Layout όπως βλέπουμε παρακάτω στις εικόνες 29, 40 ανοίξαμε την activity_main μας μέσω του Visual layout Editor του Android Studio ώστε να την σχεδιάσουμε.

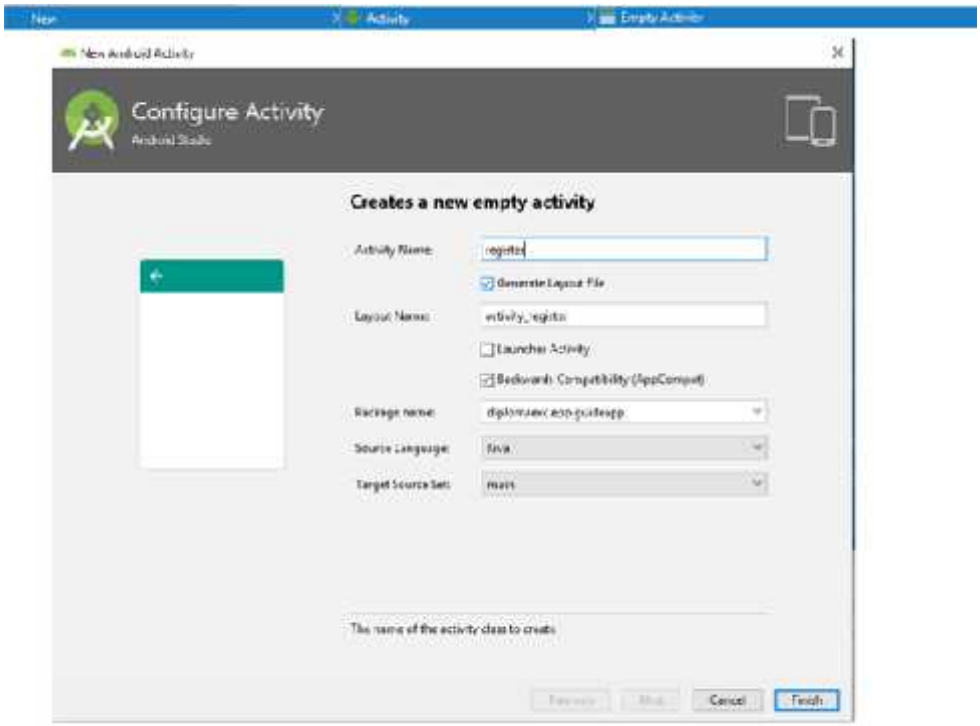


[ΕΙΚΟΝΑ 29 | ΑΡΧΕΙΑ ΠΡΟΤΖΕΚΤ]



[ΕΙΚΟΝΑ 30 | ACTIVITY MAIN]

2) Αφού τελειώσαμε τον σχεδιασμό της Activity Main, το επόμενο βήμα μας ήταν να δημιουργήσουμε νέα κλάση όπως φαίνεται παρακάτω:



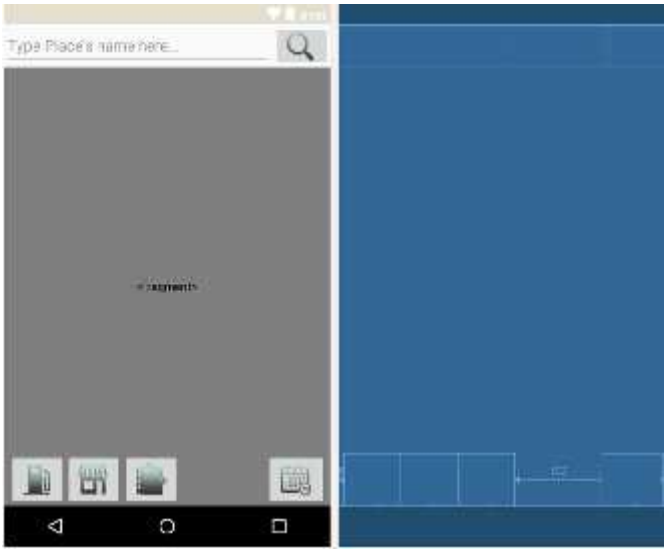
[ΕΙΚΟΝΑ 31 | ΠΑΡΑΔΕΙΓΜΑ ΣΧΕΔΙΑΣΜΟΥ ΝΕΑΣ ΚΛΑΣΗΣ]

Η κλάση αυτή είναι η Register την οποία σχεδιάσαμε στο Visual layout Editor όπως κάναμε στο Βήμα 1 με σκοπό ένας χρήστης να εγγράφεται.



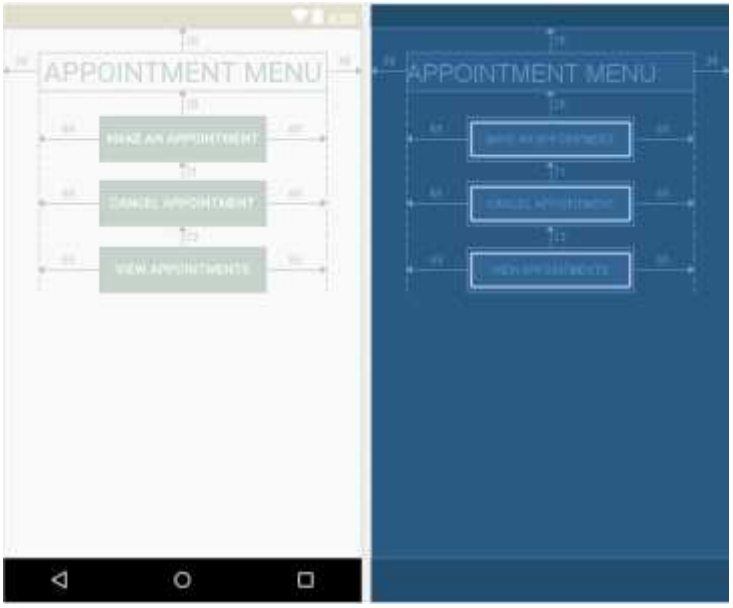
[EIKONA 32 | ACTIVITY REGISTER]

3) Μετά δημιουργήσαμε την κλάση Googleusermaps την οποία σχεδιάσαμε στο Visual layout Editor όπως βλέπουμε στην εικόνα 33 με σκοπό ο χρήστης να εισάγεται στο κεντρικό Μενού επιλογών της εφαρμογής μας όπου θα μπορεί να αναζητήσει τα πλησιέστερα βενζινάδικα, σουπερμάρκετ ή κινηματογράφους ή να αναζητήσει κάτι άλλο που επιθυμεί. Επίσης, υπάρχει η επιλογή η οποία θα τον οδηγεί στο Μενού των ραντεβού όπως θα δούμε παρακάτω.



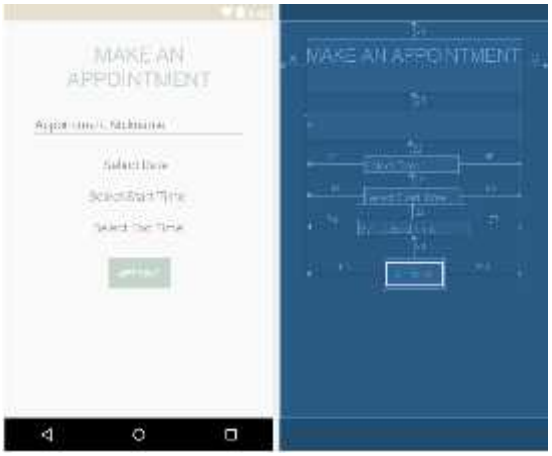
[EIKONA 33 | ACTIVITY GOOGLEUSERMAPS]

4) Στην συνέχεια δημιουργήσαμε την κλάση appointment_menu την οποία σχεδιάσαμε στο Visual layout Editor όπως βλέπουμε στην εικόνα 34 με σκοπό ο χρήστης να έχει την επιλογή να κλείσει ή να ακυρώσει κάποιο ραντεβού ή να δει τι ραντεβού έχει.

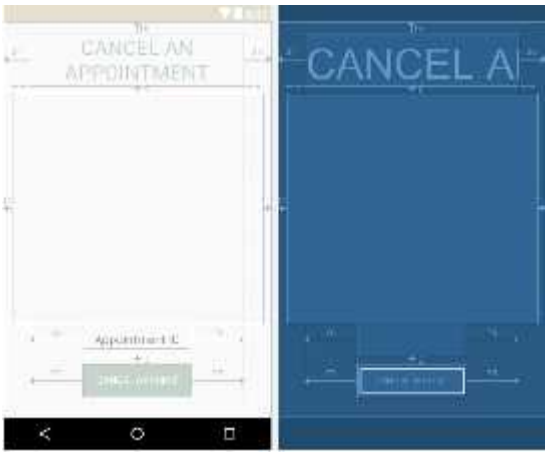


[EIKONA 34 | ACTIVITY APPOINTMENTMENU]

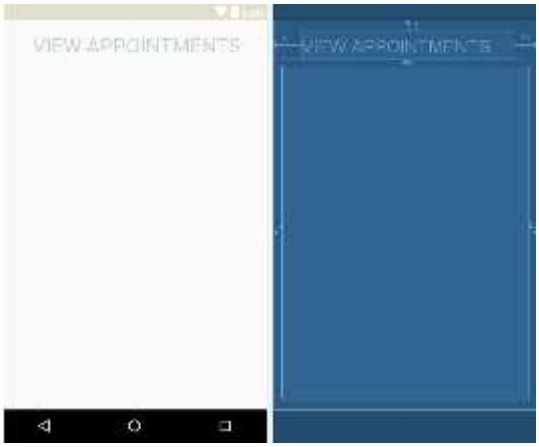
5) Τέλος, δημιουργήσαμε μία κλάση για την κάθε επιλογή που έχει ο χρήστης στο Βήμα 4 και τις σχεδιάσαμε στο Visual layout Editor. Αυτές οι κλάσεις είναι κατά σειρά η make_appointment, η cancel_appointment και η view_appointment όπως φαίνονται στις παρακάτω αντίστοιχες εικόνες 35, 36, 37.



[EIKONA 35 | ACTIVITY MAKE APPOINTMENT]



[EIKONA 36 | ACTIVITY CANCEL APPOINTMENT]



[ΕΙΚΟΝΑ 37 | ACTIVITY VIEW APPOINTMENT]

*Όλα τα παραπάνω βήματα σχεδιάστηκαν στο Visual layout Editor.
Ο Visual layout Editor του Android Studio μας δίνει τη δυνατότητα να δημιουργούμε layouts σχεδιάζοντας με drag and drop components στην οθόνη, αντί να γράφουμε με το χέρι το layout XML.

4.3 Υλοποίηση

Μετά το τέλος του σχεδιασμού της διεπαφής της εφαρμογής μας όπως είδαμε παραπάνω, ξεκινήσαμε τη φάση της ανάπτυξης της εφαρμογής σε προγραμματιστικό επίπεδο σε Java. Αρχικά, έπρεπε να υλοποιήσουμε την φάση εγγραφής του χρήστη δηλαδή να μπορεί να δημιουργήσει έναν προσωπικό λογαριασμό συμπληρώνοντας κάποια απαραίτητα πεδία τα οποία είναι το όνομα και το επώνυμο του, το email, η διεύθυνση, ο αριθμός τηλεφώνου, ένα username ένα password και μια επιβεβαίωση password καθώς και ο αριθμός της πιστωτικής του κάρτας. Για όλα τα παραπάνω πεδία, προτού εισαχθούν και αποθηκευτούν τα στοιχεία του χρήστη, έπρεπε να δημιουργήσουμε μια κλάση στην περίπτωση μας την DatabaseHelper, η οποία μας βοήθησε να δημιουργήσουμε μια βάση δεδομένων μαζί με πίνακες ώστε να μπορούμε να αποθηκεύσουμε τα στοιχεία που επιθυμούμε. Αφού δημιουργήσαμε την κλάση «βοηθό» για την βάση μας (όπως φαίνεται στο Παράρτημα 1), συνεχίσαμε κάνοντας έλεγχο εγκυρότητας των στοιχείων αυτών, ώστε να είναι μοναδικά για τον χρήστη τα στοιχεία αυτά και να μην υπάρχει η περίπτωση της διπλοτυπίας τους. Όλα αυτά τα πεδία που συμπληρώνει ο χρήστης, αποθηκεύονται σε μία βάση δεδομένων, εφόσον περάσουν τους απαραίτητους ελέγχους εγκυρότητας που έχουμε ορίσει προγραμματιστικά (όπως φαίνεται στο Παράρτημα 2). Προκειμένου, να μεταφερθεί ένας χρήστης από την Activity Main που είναι το κεντρικό μενού της εφαρμογής μας, στο Μενού της Εγγραφής του χρήστη θα πρέπει να πατήσει το κουμπί Register το οποίο έχει υλοποιηθεί προγραμματιστικά (όπως φαίνεται στο Παράρτημα 3).

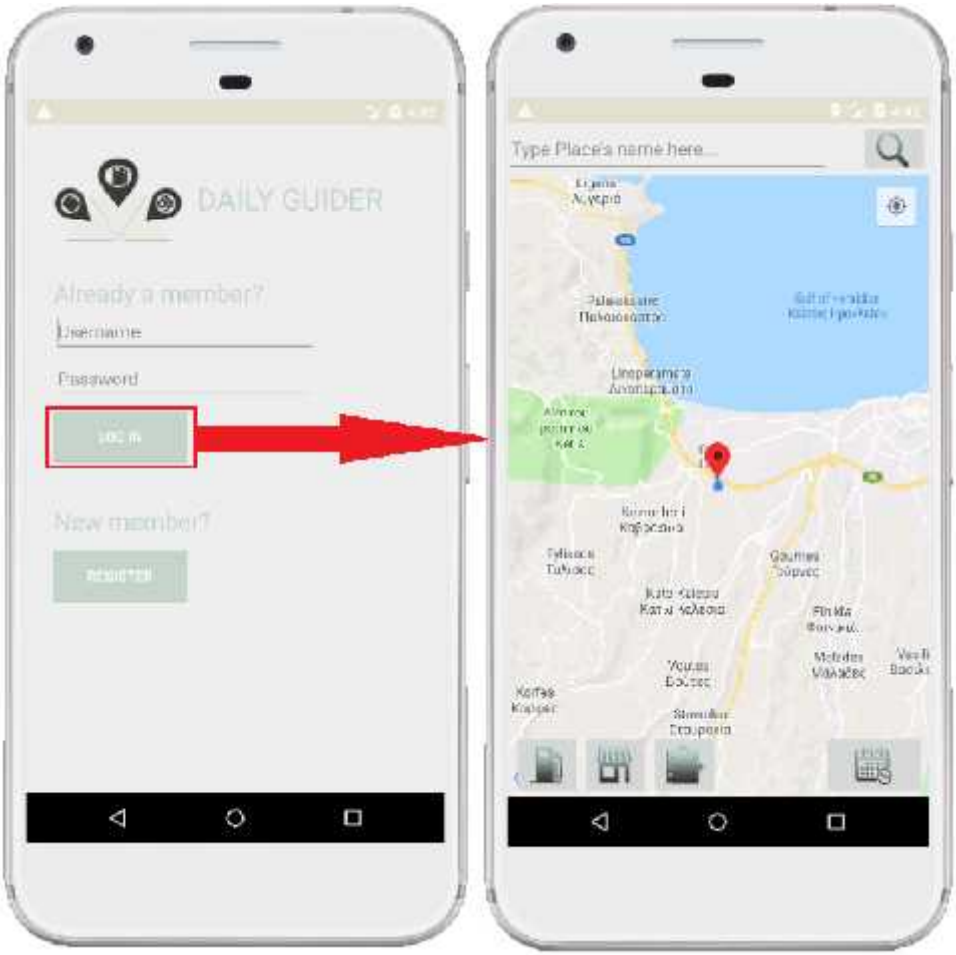
Ο σκοπός που επιτελεί ο κώδικας του Παραρτήματος 3, είναι να φέρνει στο παρασκήνιο ένα άλλο Activity που στην παρούσα φάση είναι το Activity Register, όπως φαίνεται στην εικόνα 38.



[ΕΙΚΟΝΑ 38 | MAIN TO REGISTER ACTIVITY]

Εφόσον γίνει επιτυχής εγγραφή του χρήστη, τότε θα εμφανιστεί ένα αντίστοιχο μήνυμα «You have been successfully Registered!». Στην συνέχεια, ο χρήστης θα πρέπει να πατήσει το κουμπί Cancel το οποίο έχει υλοποιηθεί προγραμματιστικά, (όπως φαίνεται στο Παράρτημα 4) με σκοπό να τελειώνει τη διαδικασία που του έχει ανατεθεί και να επιστρέφει τον χρήστη στο κεντρικό Μενού. Το επόμενο βήμα είναι ο προγραμματισμός του κουμπιού «Log in», σκοπός του είναι να ελέγχει τα δύο πεδία που έχει συμπληρώσει ο χρήστης (Username, Password), δηλαδή να επαληθεύει ότι τα στοιχεία αυτά υπάρχουν μέσα στην βάση δεδομένων και τότε και μόνο τότε να τον εισάγει στο Μενού των λειτουργιών της εφαρμογής μας (όπως φαίνεται στην εικόνα 39 και στο Παράρτημα 5). Αν τα στοιχεία είναι έγκυρα, τότε θα εμφανίζει το μήνυμα «You have been successfuly logged in » μαζί με το όνομα του χρήστη και θα εμφανίζει στο παρασκήνιο το Activity Googleusermaps αλλά αυτή την φορά με ένα διαφορετικό τρόπο γιατί

θα θέλουμε να μεταφέρουμε μαζί και το username του χρήστη ο οποίος εισήχθη (όπως φαίνεται στο Παράρτημα 5), ενώ αν δεν είναι, τότε θα εμφανίζει το μήνυμα «Error! Check your Username and Password.».



[ΕΙΚΟΝΑ 39 | MAIN TO GOOGLEUSERMAPS ACTIVITY]

Ενώ έχουμε εισαχθεί στο κεντρικό Μενού Επιλογών της εφαρμογής μας, πρέπει να προγραμματίσουμε την κλάση μας, ώστε στο παρασκήνιο να δίνεται άδεια στο να μπορεί να χρησιμοποιήσει τα Google Services, ACCESS_COARSE_LOCATION, INTERNET, ACCESS_NETWORK_STATE και το GPS. Επιπλέον, αφού έχουν δοθεί αυτές οι άδειες, η εφαρμογή μας θα βρίσκει την ακριβή τοποθεσία του χρήστη με την εμφάνιση ενός marker πάνω στον χάρτη, μέσω του GPS του κινητού στο οποίο έχει δοθεί η άδεια, και θα εμφανίζεται στο object του Google maps που έχουμε βάλει στο Activity Googleusermaps. Αυτό έχει υλοποιηθεί προγραμματιστικά μέσω συναρτήσεων και μεθόδων που προσφέρει η ίδια η Google για τις υπηρεσίες του Google maps που χρησιμοποιούνται για προγραμματιστικό σκοπό, όπως στην περίπτωση μας, στο Android Studio (όπως φαίνεται στην εικόνα 39 και στο Παράρτημα 6).

Το επόμενο βήμα ήταν να προγραμματίσουμε την λειτουργία κάθε κουμπιού. Αρχικά, προγραμματίσαμε το κουμπί της Αναζήτησης διότι είχε την πιο απλή υλοποίηση, (όπως φαίνεται στο Παράρτημα 6) σκοπός του οποίου είναι στην αρχή να «καθαρίζει» τα markers του πίνακα και στην συνέχεια αναζητεί την τοποθεσία που εισάγει ο χρήστης στο πεδίο Εισαγωγής και του εμφανίζει μέσω ενός νέου marker την τοποθεσία που αναζήτησε ο χρήστης. Μετά από αυτό, υπάρχουν άλλες 3 επιλογές οι οποίες είναι η εύρεση των πλησιέστερων βενζινάδικων, σουπερμάρκετ και κινηματογράφων και οι 3 επιλογές αυτές υλοποιούνται με παρόμοιο τρόπο με βήματα τα οποία είναι τα εξής (όπως φαίνεται στο Παράρτημα 7):

- Να καθαρίζει τον χάρτη από όλα τα markers,
- Να εισάγει ένα marker πάνω στον χάρτη ο οποίος δείχνει την τοποθεσία του χρήστη,
- Να αποθηκεύει σε μια μεταβλητή string ένα url το οποίο έχει δημιουργηθεί καλώντας μία συνάρτηση την getUrl η οποία παίρνει ως παραμέτρους τις συντεταγμένες του χρήστη αλλά και πιο κουμπί πάτησε για να αναζητηθεί η επιλογή που αντιστοιχεί στο κουμπί. Αυτή η συνάρτηση όπως είπαμε φτιάχνει ένα string url το οποίο παρέχεται από τα APIs της Google και πιο συγκεκριμένα, το API place, nearby search το οποίο χρειάζεται να περιέχει τις συντεταγμένες του χρήστη, την εμβέλεια στην οποία θα αναζητήσει, το τι θα αναζητήσει, να του πούμε ότι οι αισθητήρες μας είναι ανοιχτοί αλλά και το μοναδικό κλειδί που δίνει η Google σε κάθε χρήστη της για να χρησιμοποιήσει τα APIs της.
- Μετά τον γυρισμό του url από την συνάρτηση getUrl αποθηκεύουμε σε έναν πίνακα 6 θέσεων τα εξής στοιχεία: το map που δημιουργήσαμε, το url που πήραμε, τις συντεταγμένες του χρήστη οι οποίες χωρίζονται σε δύο (το latitude και το longitude), τι πατήθηκε, αλλά και το username του χρήστη.
- Όλα αυτά μεταφέρονται και εκτελούνται μέσα σε μια άλλη ασύγχρονη κλάση της οποίας πρέπει να φτιάξουμε ένα Instance πριν την χρησιμοποιήσουμε τη getNearbyPlaces, η οποία θα αναλυθεί παρακάτω.

Αρχικά, η getNearbyPlaces αποθηκεύει τα στοιχεία που της στείλαμε από την κλάση Googleusermaps σε τοπικές μεταβλητές και στην συνέχεια καλεί την τοπική συνάρτηση downloadDirectionsUrl η οποία παίρνει σαν παράμετρο το url που στείλαμε από την κλάση Googleusermaps. Με την σειρά της η downloadDirectionsUrl φτιάχνει ένα instance της εξωτερικής κλάσης downloadUrl της οποίας η δουλειά είναι μέσω διαφόρων μεθόδων, (όπως φαίνεται στο Παράρτημα 8) να διαβάζει και να μας επιστρέφει τα περιεχόμενα του url που της δώσαμε. Στην συνέχεια, δημιουργούμε ένα νέο instance της κλάσης dataParser της οποίας η δουλειά είναι να χωρίζει τα δεδομένα όπως θα τις ορίσουμε εμείς με διάφορους τρόπους. Τα περιεχόμενα που θα μας χωρίσει, θα αποθηκευτούν σε μια λίστα η οποία θα έχει μια θέση για κάθε ένα από τις τοποθεσίες που βρίσκει θα διαθέτει μία θέση. Για κάθε θέση της λίστας αυτής, θα αντλούμε τα δεδομένα (όνομα του μέρους, μοναδικό reference key, latitude και longitude), θα τα αποθηκεύουμε σε μεταβλητές και θα ξανά καλούμε την getDirectionsUrl με αυτή την φορά

να έχει ως ορίσματα το latitude και το longitude που πήραμε με σκοπό να πάρουμε το url και να το δώσουμε σαν όρισμα στην downloadDirectionsUrl με σκοπό να πάρουμε τα περιεχόμενα αυτού. Στην συνέχεια, θα καλέσουμε μία συνάρτηση της dataParser, την parseDurDist η οποία θα παίρνει σαν όρισμα τα περιεχόμενα που πήραμε από την downloadDirectionsUrl με σκοπό να μας δώσει την απόσταση και την ώρα από την τοποθεσία μας προς την τοποθεσία του καταστήματος. Αυτά θα αποθηκεύονται σε δύο μεταβλητές αντίστοιχα.

Το επόμενο βήμα είναι να παίρνουμε τα στοιχεία του προορισμού με την κοντινότερη διαδρομή με βάση τον χρόνο και να αποθηκεύουμε τα περιεχόμενα αυτουνού σε ξεχωριστές μεταβλητές, υπόψιν να παρθεί ότι εάν η κοντινέστερη διαδρομή της οποίας έχουμε πάρει τις μεταβλητές έχει ίδια διαδρομή χρόνου με μία άλλη τότε θα διαλέξουμε αυτή που έχει την μικρότερη διαδρομή σε απόσταση. Εφόσον παρθούν αυτές οι πληροφορίες, τότε θα εισάγεται στον χάρτη ένας marker στην τοποθεσία του κάθε καταστήματος ο οποίος θα γράφει το όνομα, την διεύθυνση, αλλά και την διαδρομή σε χρόνο και σε απόσταση. Αφού εμφανίσουμε όλα τα πλησιέστερα μέρη και πάρουμε τα στοιχεία του πλησιέστερου όλων ας πούμε x, μέσω της κλάσης downloadDirectionsUrl που αυτή την φορά θα πάρει ως όρισμα το url του x. Στην συνέχεια, θα καλέσουμε μία συνάρτηση την parseHours της κλάσης dataParser η οποία θα επιστρέφει σε μία λίστα τις ώρες που είναι ανοιχτό ή κλειστό το κατάστημα.

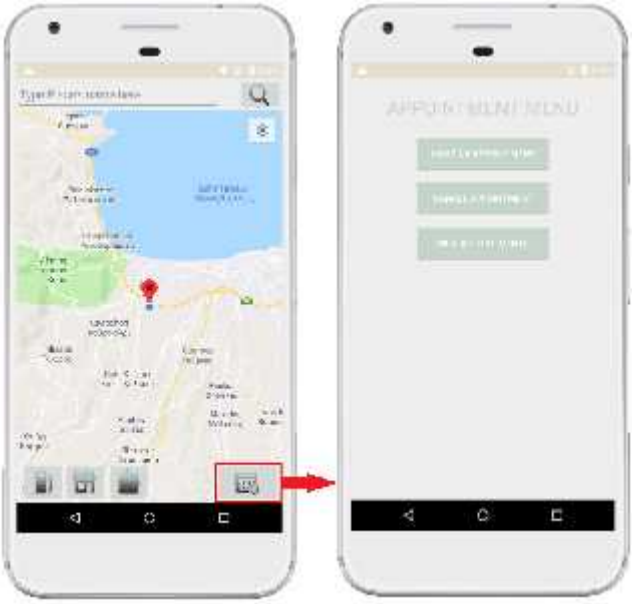
Το επόμενο βήμα μας αφού πήραμε τις διαθέσιμες ώρες του x καταστήματος, είναι να βεβαιωθούμε ότι είναι ανοιχτό την ώρα που το χρειάζεται ο χρήστης. Για να γίνει αυτό, συγκρίνουμε την ημερομηνία και την ώρα του κινητού με αυτή του x καταστήματος δηλαδή αν είναι ίδιες μέρες της εβδομάδας (π.χ. Τρίτη και Τρίτη), τότε να πάρει τις ώρες για αυτήν την ημέρα και να ελέγχει αν ο χρήστης έχει την δυνατότητα να πάει σε σχέση με το χρόνο που έχει διαθέσιμο αυτός αλλά και το κατάστημα. Εάν το x κατάστημα δεν μας δώσει κάποιο διαθέσιμο ωράριο σαν αποτέλεσμα, τότε το πρόγραμμα μας θα ψάξει για κάποιο άλλο κατάστημα, αλλά πριν το κάνει θα βάλει το x αποτέλεσμα σε μία λίστα η οποία αποσκοπεί στον να μην ξανά πάρουμε αυτό σαν αποτέλεσμα. Εφόσον τελειώσει αυτή η διαδικασία και βρούμε το κατάλληλο αποτέλεσμα, τότε θα καλείται μία συνάρτηση η parseDirections της κλάσης dataParser η οποία θα δίνει το αποτέλεσμα της στην τοπική συνάρτηση displayDirection της οποίας ο ρόλος είναι να «ζωγραφίσει» την πλησιέστερη διαδρομή μέχρι τον προορισμό μας. Τέλος, θα δίνουμε ένα αποτέλεσμα Ψευδές σε μία μεταβλητή με σκοπό να σημάνουμε ότι έχουμε βρει έναν προορισμό σαν αποτέλεσμα (ή ότι η επανάληψη μας τελειώνει).

Όλα αυτά επιτεύχθηκαν όπως φαίνεται στο Παράρτημα 8 και το αποτέλεσμα φαίνεται στην εικόνα 40.



[ΕΙΚΟΝΑ 40 | ΑΠΟΤΕΛΕΣΜΑ ΑΝΑΖΗΤΗΣΗΣ]

Ολοκληρώνοντας, η τελευταία υλοποίηση που έπρεπε να ολοκληρώσουμε, ήταν αυτή του κουμπιού των ραντεβού η οποία θα έχει σκοπό να μεταφέρει τον χρήστη σε ένα άλλο Μενού στο οποίο θα έχει την επιλογή να κλείνει ραντεβού, να ακυρώνει ραντεβού αλλά και να εμφανίζει τα ραντεβού του όπως φαίνεται στην εικόνα 41.



[ΕΙΚΟΝΑ 41 | ΜΕΝΟΥ ΡΑΝΤΕΒΟΥ]

Στην πρώτη περίπτωση που ο χρήστης επιθυμεί να κλείσει ένα ραντεβού, θα πρέπει να συμπληρώσει κάποια πεδία τα οποία είναι: το όνομα του ραντεβού, η ημερομηνία και ώρα έναρξης, αλλά και η ώρα λήξης του ραντεβού, όπως φαίνεται στην εικόνα 42. Για κάθε ένα από αυτά τα πεδία που συμπληρώνει ο χρήστης, γίνεται έλεγχος εγκυρότητας. Αναλυτικότερα, στο πεδίο της ημερομηνίας πρέπει να εισαχθεί η ημερομηνία μεγαλύτερη ή ίση της τωρινής ημερομηνίας που έχουμε. Η ώρα έναρξης πρέπει να είναι μεγαλύτερη ή ίση της τωρινής ώρας που έχουμε αλλά και μικρότερη από την ώρα λήξης. Η ώρα λήξης πρέπει να είναι μεγαλύτερη από την ώρα έναρξης αλλά και το συνολικό διάστημα αυτών των 2 ωρών δεν πρέπει να συμπίπτει με άλλο ραντεβού της ίδιας ημερομηνίας. Τέλος, εάν όλα τα πεδία έχουν συμπληρωθεί σωστά, τότε τα περιεχόμενα αυτών μαζί με ένα μοναδικό αυτόματο αυξανόμενο ID θα αποθηκεύονται στη βάση δεδομένων.



[ΕΙΚΟΝΑ 42 | ΚΛΕΙΣΙΜΟ ΡΑΝΤΕΒΟΥ]

Στη δεύτερη περίπτωση που ο χρήστης επιθυμεί να ακυρώσει ένα ραντεβού, θα πρέπει να εισάγει στο πεδίο Appointment ID, το ID του ραντεβού που θέλει να διαγράψει, όπως φαίνεται στην εικόνα 43. Αυτή η διαγραφή γίνεται μέσα στην βάση δεδομένων. Εάν υπάρχουν ραντεβού, τότε θα εμφανίζονται σε μία λίστα, ενώ εάν δεν υπάρχουν ραντεβού, τότε θα εμφανίζεται αντίστοιχο μήνυμα. Επιπλέον, έχουμε προγραμματίσει την εφαρμογή μας έτσι ώστε όταν συνδέεται ένας χρήστης να γίνεται έλεγχος όλων των ραντεβού του από την βάση δεδομένων και να διαγράφονται αυτά που η πραγματοποίηση τους είναι μικρότερη της τωρινής ημερομηνίας και ώρας μας.



[ΕΙΚΟΝΑ 43 | ΔΙΑΓΡΑΦΗ ΡΑΝΤΕΒΟΥ]

Στην τρίτη περίπτωση, θα εμφανίζονται όλα τα ραντεβού που έχει ο χρήστης αλλιώς αν δεν έχει ραντεβού, τότε θα εμφανίζεται αντίστοιχο μήνυμα, όπως φαίνεται στην εικόνα 44.



[ΕΙΚΟΝΑ 44 | ΕΜΦΑΝΙΣΗ ΡΑΝΤΕΒΟΥ]

Τέλος, για την ορθή και σωστή χρήση της εφαρμογής μας έπρεπε να επιστρέψουμε στην κλάση getNearbyPlaces και να κάνουμε τους κατάλληλους ελέγχους ώστε εάν ο χρήστης επιθυμεί να πάει κάπου αλλά ταυτόχρονα σε αυτό το διάστημα της μετακίνησης του έχει κανονισμένο ραντεβού το οποίο ενδεχομένως να έχει ξεχάσει, τότε θα του εμφανίζεται κατάλληλο μήνυμα (όπως φαίνεται στο Παράρτημα 8).

5. Αποτελέσματα

5.1 Συμπεράσματα

Οι βασικοί στόχοι ανάπτυξης της εφαρμογής επιτεύχθηκαν. Στην εφαρμογή μας αποθηκεύονται σημαντικές πληροφορίες που θα αναπτυχθούν παρακάτω καθώς και εκπληρεί πολύ σημαντικές λειτουργίες. Ο λόγος για τον οποίο αναπτύξαμε την εφαρμογή μας στο Android Studio και έτσι θα λειτουργείται σε Android συσκευές, είναι λόγου του ποσοστού χρήσης του λειτουργικού συστήματος αυτού σε σχέση με τα υπόλοιπα λειτουργικά συστήματα που υπάρχουν στην αγορά. Επομένως, είναι μια εύχρηστη εφαρμογή που καλύπτει πολλές ανάγκες τον χρηστών ώστε η καθημερινότητα τους να γίνεται πιο απλή και εύκολη χωρίς άγχος και δυσχέρεια.

Αναλυτικότερα, χρειάστηκαν για να παρθούν και να αποθηκευτούν οι παρακάτω πληροφορίες για:

- τα βενζινάδικα (δηλαδή ωράριο και διεύθυνση).
- τα σουπερμάρκετ (δηλαδή ωράριο και διεύθυνση).
- τους κινηματογράφους (δηλαδή διεύθυνση, ωράριο).
- τον χάρτη της πόλης (δηλαδή οι αποστάσεις από την μια περιοχή στην άλλη όπως και ο χρόνος που χρειάζεται για τις μετακινήσεις που χωρίζεται όταν υπάρχει ή όχι κυκλοφορικό πρόβλημα). Επίσης, δηλώνονται και οι ώρες 8:00-9:00 και 13:00-15:00 ως κυκλοφοριακή συμφόρηση οπότε οι μετακινήσεις θα είναι πιο αργές.
- τον κάθε χρήστη του συστήματος (δηλαδή όνομα, διεύθυνση, τηλέφωνο, αριθμό πιστωτικής και μοναδικό κωδικό που θα ανατίθεται στον πελάτη αυτόματα). Ακόμη, ο χρήστης θα έχει ένα username και ένα password.

Στη συνέχεια, το σύστημα υποστηρίζει τις λειτουργίες που είναι:

- η εγγραφή ενός νέου χρήστη. Ο χρήστης θα δίνει όλα τα απαραίτητα στοιχεία και θα επιλέγει ένα δικό του username και password. Το σύστημα θα ελέγχει αν είναι μοναδικό το username και αν δεν είναι θα ζητάει από τον χρήστη να δώσει νέο κωδικό.
- εισαγωγή και ακύρωση ραντεβού κάποιου χρήστη. Όταν ο χρήστης εισάγει ένα νέο ραντεβού θα γίνεται έλεγχος εάν οι υπόλοιπες υποχρεώσεις του χρήστη επιτρέπουν την πραγματοποίηση του. Εάν, παραδείγματος χάρη, έχει άλλο ραντεβού την ίδια ώρα ή αν έχει ραντεβού που τελειώνει λίγο πριν άλλα λόγω της απόστασης ή της κυκλοφοριακής συμφόρησης δεν προλαβαίνει να πάει στο νέο ραντεβού, τότε θα του εμφανίζεται αντίστοιχο μήνυμα ότι δεν προλαβαίνει.
- ο χρήστης μπορεί να ζητήσει να πάει σε κάποιο βενζινάδικο. Το σύστημα ανάλογα με την περιοχή που βρίσκεται θα τον δρομολογεί στο πιο κοντινό βενζινάδικο της περιοχής

στο οποίο θα προλαβαίνει να πάει πριν αυτό κλείσει. Συνεπακόλουθα, αυτό σημαίνει ότι δεν είναι απαραίτητο να τον στείλει στο πιο κοντινό αν αυτό θα κλείσει μέχρι να πάει εκεί ή αν είναι ήδη κλειστό. Ακόμη, λαμβάνονται υπόψη και οι επόμενες υποχρεώσεις του χρήστη. Εάν από το πιο κοντινό βενζινάδικο το οποίο δεν θα έχει κλείσει μέχρι να πάει σε αυτό ο χρήστης, η απόσταση από εκεί μέχρι το επόμενο ραντεβού του χρήστη είναι μεγάλη και δεν προλαβαίνει να πάει ο χρήστης, τότε το σύστημα θα τον δρομολογεί σε βενζινάδικο που είναι στο δρόμο του. Στην περίπτωση που δεν υπάρχει τέτοιο βενζινάδικο, τότε θα τον δρομολογεί κάπου αλλού εμφανίζοντας το αντίστοιχο μήνυμα.

- ο χρήστης μπορεί να ζητήσει να πάει σε κάποιο κινηματογράφο. Η δρομολόγηση, λοιπόν, θα γίνεται με κριτήρια αν προλαβαίνει ο χρήστης να πάει στο πιο κοντινό κινηματογράφο.
- ο χρήστης μπορεί να ζητήσει να πάει σε κάποιο σουπερμάρκετ. Η δρομολόγηση γίνεται όπως παραπάνω και στα βενζινάδικα με τη διαφορά ότι θα υπολογίζεται και ένα σαραντάλεπτο για ψώνια. Εάν ο χρήστης δεν προλαβαίνει, τότε θα του εμφανίζεται κατάλληλο μήνυμα.

5.2 Μελλοντική Εργασία και Επεκτάσεις

Παρόλα αυτά όπως κάθε εφαρμογή, έτσι, και για τη δική μας υπάρχουν τρόποι βελτίωσης και θα αναλυθούν παρακάτω:

- <u>Πρόσθεση επιπλέων δραστηριοτήτων.</u>
  Δηλαδή, υπάρχει η δυνατότητα πρόσθεσης και άλλων δραστηριοτήτων δρομολόγησης των χρηστών εκτός από κινηματογράφο, βενζινάδικο και σουπερμάρκετ. Για να προστεθούν οι επιπλέον δραστηριότητες, χρειάζεται να γίνει έρευνα ως προς την ανάγκη των χρηστών χρησιμοποιώντας πληροφορίες και από καταστήματα στην κοινωνία της περιοχής που είναι.

- <u>Δυνατότητα αναζήτησης αξιολογήσεων για την κάθε δραστηριότητα που θέλει.</u>
  Δηλαδή ο χρήστης όταν θέλει να πάει σε ένα κινηματογράφο, βενζινάδικο ή σουπερμάρκετ, θα μπορεί να μπαίνει και να αναζητεί σε αυτό που του προτείνει η εφαρμογή μας και είναι το πιο κοντινό, εάν η αξιολόγηση από τους άλλους χρήστες διαβάζοντας τα σχόλια τους ως προς την εξυπηρέτηση, την ποιότητα, την τιμή ή οτιδήποτε άλλο είναι καλά ως προς αυτόν για να πάει και αν μείνει ευχαριστημένος να ξανά πηγαίνει και να είναι τακτικός πελάτης αλλιώς να πάει στην αμέσως επόμενη ή παρεπόμενη επιλογή που του δίνει η εφαρμογή. Έτσι, είναι σημαντική η ευχαρίστηση

του χρήστη και ως προς τις υπηρεσίες που προσφέρει η κάθε δραστηριότητα σε κάθε μέρος.

- Επέκταση της εφαρμογής σε άλλες συσκευές πέρα από τις Android.
  Δηλαδή, θα μπορούσε η εφαρμογή μας να μεταφερθεί και σε άλλες συσκευές εκτός λογισμικό Android όπως σε συσκευές Windows, iOS και άλλες.



[ΕΙΚΟΝΑ 45 | ΓΡΑΦΗΜΑ ΧΡΟΝΟΛΟΓΙΩΝ ΛΕΙΤΟΥΡΓΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ 2017 - 2019]

Όπως βλέπουμε (στην εικόνα 45) από το γράφημα χρονολογιών 2017-2019, που απεικονίζει τα αποτελέσματα έρευνας και δείχνει τα στατιστικά σε ποσοστά ανάλογα με τη χρήση λογισμικών συσκευών, πρέπει να υπογραμμιστεί ότι το ποσοστό χρηστών με λειτουργικό Android είναι εμφανέστερα μεγαλύτερο σε σχέση με τα υπόλοιπα. Παρόλα αυτά έχουμε καλά ποσοστά και στα υπόλοιπα λογισμικά οπότε θα ήταν ωφέλιμη η επέκταση της εφαρμογής μας και σε άλλες συσκευές πέρα από τις Android ώστε να έχουν και άλλοι χρήστες με άλλα λογισμικά τη δυνατότητα να την χρησιμοποιήσουν.

Πίνακες

Πίνακες εικόνων

ΑΡΙΘΜΟΣ ΕΙΚΟΝΑΣ	ΤΙΤΛΟΣ ΕΙΚΟΝΑΣ	ΣΕΛΙΔΑ
1	ΔΙΑΓΡΑΜΜΑ ΑΠΑΙΤΗΣΗΣ ΧΡΟΝΟΥ - ΔΡΑΣΤΗΡΙΟΤΗΤΑΣ	7
2	ΠΙΝΑΚΑΣ ΣΗΜΑΝΤΙΚΟΤΗΤΑΣ - ΕΠΕΙΓΟΝΤΟΣ	7
3	ΠΙΝΑΚΑΣ ΣΗΜΑΝΤΙΚΟΤΗΤΑΣ - ΕΠΕΙΓΟΝΤΟΣ	9
4	ΛΟΓΟΤΥΠΟ ΠΡΟΓΡΑΜΜΑΤΟΣ ANDROID STUDIO	10
5	ΕΙΚΟΝΙΚΗ ΣΥΣΚΕΥΗ ANDROID (EMULATOR)	11
6	ΔΕΙΓΜΑ ΔΗΜΙΟΥΡΓΙΑΣ ΕΝΩΣ ΝΕΟΥ ΠΡΟΤΖΕΚΤ (1)	11
7	ΔΕΙΓΜΑ ΔΗΜΙΟΥΡΓΙΑΣ ΕΝΩΣ ΝΕΟΥ ΠΡΟΤΖΕΚΤ (2)	12
8	ΔΕΙΓΜΑ ΔΗΜΙΟΥΡΓΙΑΣ ΕΝΩΣ ΝΕΟΥ ΠΡΟΤΖΕΚΤ (3)	12
9	ΔΕΙΓΜΑ ΔΗΜΙΟΥΡΓΙΑΣ ΕΝΩΣ ΝΕΟΥ ΠΡΟΤΖΕΚΤ (4)	13
10	ΤΟ ΚΕΝΤΡΙΚΟ ΜΕΝΟΥ ΤΗΣ ΕΦΑΡΜΟΓΗΣ ΜΑΣ ΑΠΟ ΤΟ VISUAL LAYOUT EDITOR	14
11	ΤΟ ΜΕΝΟΥ ΕΓΓΡΑΦΗΣ ΤΗΣ ΕΦΑΡΜΟΓΗΣ ΜΑΣ ΑΠΟ ΤΟ VISUAL LAYOUT EDITOR	15
12	VIEWGROUP	16
13	ΠΑΡΑΔΕΙΓΜΑ ΚΩΔΙΚΑ XML	17
14	EMULATOR MAIN	17
15	EMULATOR REGISTER	17
16	EMULATOR MAP	18
17	JAVATUTORIAL ΚΛΑΣΕΙΣ	19
18	JAVATUTORIAL CONSTRUCTOR	19
19	JAVATUTORIAL ΚΛΗΡΟΝΟΜΙΚΟΤΗΤΑ	20
20	SQLITE DATABASE OBJECT	20
21	ADOBE PHOTOSHOP LOGO	22
22	ΣΤΟΧΟΙ ΟΛΟΚΛΗΡΩΣΗΣ ΕΦΑΡΜΟΓΗΣ	23
23	ΛΟΓΟΤΥΠΑ ΤΗΣ ΕΦΑΡΜΟΓΗΣ	25
24	ΕΙΚΟΝΙΔΙΟ ΒΕΝΖΙΝΑΔΙΚΟΥ	25
25	ΕΙΚΟΝΙΔΙΟ SUPERMARKET	25
26	ΕΙΚΟΝΙΔΙΟ ΚΙΝΗΜΑΤΟΓΡΑΦΟΥ	26
27	ΕΙΚΟΝΙΔΙΟ ΜΕΝΟΥ ΡΑΝΤΕΒΟΥ	26
28	ΕΙΚΟΝΙΔΙΟ ΑΝΑΖΗΤΗΣΗΣ ΤΟΠΟΘΕΣΙΑΣ	26
29	ΑΡΧΕΙΑ ΠΡΟΤΖΕΚΤ	26
30	ACTIVITY MAIN	27
31	ΠΑΡΑΔΕΙΓΜΑ ΣΧΕΔΙΑΣΜΟΥ ΝΕΑΣ ΚΛΑΣΗΣ	27
32	ACTIVITY REGISTER	28
33	ACTIVITY GOOGLEUSERMAPS	28
34	ACTIVITY APPOINTMENTMENU	29
35	ACTIVITY MAKE APPOINTMENT	29

36	ACTIVITY CANCEL APPOINTMENT	29
37	ACTIVITY VIEW APPOINTMENT	30
38	MAIN TO REGISTER ACTIVITY	31
39	MAIN TO GOOGLEUSERMAPS ACTIVITY	32
40	ΑΠΟΤΕΛΕΣΜΑ ΑΝΑΖΗΤΗΣΗΣ	35
41	ΜΕΝΟΥ ΡΑΝΤΕΒΟΥ	35
42	ΚΛΕΙΣΙΜΟ ΡΑΝΤΕΒΟΥ	36
43	ΔΙΑΓΡΑΦΗ ΡΑΝΤΕΒΟΥ	37
44	ΕΜΦΑΝΙΣΗ ΡΑΝΤΕΒΟΥ	37
45	ΓΡΑΦΗΜΑ ΧΡΟΝΟΛΟΓΙΩΝ ΛΕΙΤΟΥΡΓΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ 2017 - 2019	41

Βιβλιογραφία

Android Studio, “Android Studio provides the fastest tools for building apps on every type of Android device”, (https://developer.android.com/studio/), (14/11/2018)

SQLite, “SQLite is a C-language library that implements a small, fast, self-contained, high-reliability, full-featured, SQL database engine”, (https://www.sqlite.org/index.html), (12/01/2018)

Ngoc Minh Tran, “Create a Database Android Application in Android Studio 3.0”, (https://dzone.com/articles/create-a-database-android-application-in-android-s), (05/12/2017)

StatCounter, “The free, online visitor stats tool”, (http://gs.statcounter.com/), (20/01/2019)

Adam Batkin, “Time comparison”, (https://stackoverflow.com/questions/2309558/time-comparison), (22/01/2010)

Angelo Fuchs, “Check if a String is in an ArrayList of Strings”, (https://stackoverflow.com/questions/10208676/check-if-a-string-is-in-an-arraylist-of-strings?fbclid=IwAR3MPdL0Dp1NAdwa275lsy-yjYCv6pWJHBURAK2VKRioi8fwKgDggo7DPuI), (18/4/2012)

Sassy Llama, “How can I include the traffic layer of google maps?”, (https://stackoverflow.com/questions/31657822/how-can-i-include-the-traffic-layer-of-google-maps?fbclid=IwAR2jIKJhF6eRs1D0cR8FRiMyO2h7u2faNNFBoGxzYgrTcE2sY4ztulm_tXU), (27/07/2015)

Hafizh Herdi, “How to create a link to google live traffic maps from my Application?”, (https://stackoverflow.com/questions/22217611/how-to-create-a-link-to-google-live-traffic-maps-from-my-application?fbclid=IwAR2A0fq77hse47WRpWnf8JevPgGJMGrwwhoQSyNQYj-h-d5lJFcToGmRX7Q), (06/03/2014)

Attaullah, “Find distance between two points on map using Google Map API V2”, (https://stackoverflow.com/questions/14394366/find-distance-between-two-points-on-map-using-google-map-api-v2?fbclid=IwAR31osQCe1CKjcULxpR07qIBohXk8jG-L8I7jNn2LU6LNHRGexU-1DOHgl0), (01/07/2015)

AlexWien, “How to calculate distance from different markers in a map and then pick up the least one”, (https://stackoverflow.com/questions/19218081/how-to-calculate-distance-from-different-markers-in-a-map-and-then-pick-up-the-l), (07/10/2013)

Bschultz, “Passing a double value through to a different class in Android Java”, (https://stackoverflow.com/questions/9588026/passing-a-double-value-through-to-a-different-class-in-android-java), (06/03/2012)

Jobbert, “How to get opening_hours using the Google Places API for Android”, (https://stackoverflow.com/questions/42676182/how-to-get-opening-hours-using-the-google-places-api-for-android), (14/11/2017)

Matt, “Extract digits from a string in Java”, (https://stackoverflow.com/questions/4030928/extract-digits-from-a-string-in-java?fbclid=IwAR1DdcSMFvgSxwgzfRhsddCzNu3W1ZSJ--7JcukiAuIWEIX08f3I_pqw2ck), (16/09/2016)

Παραρτήματα της Πτυχιακής Εργασίας

Παράρτημα Α1 : DatabaseHelper

```
package diplomaonc.app.guideapp;

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

public class DatabaseHelper extends SQLiteOpenHelper {
    //declare vars
    public static final String DATABASE_NAME="database.db";
    public static final String TABLE_NAME="register";
    public static final String TABLE_NAME2="appointments";
    public static final String COL_1="ID";
    public static final String COL_2="username";
    public static final String COL_3="password";
    public static final String COL_4="email";
    public static final String COL_5="firstName";
    public static final String COL_6="lastName";
    public static final String COL_7="address";
    public static final String COL_8="phone";
    public static final String COL_9="color";

    public DatabaseHelper(Context context) {
        super(context, DATABASE_NAME, null, 1);
    }

    //Create table
    @Override
    public void onCreate(SQLiteDatabase db) {
        db.execSQL("CREATE TABLE " +TABLE_NAME+ "(ID INTEGER PRIMARY KEY AUTOINCREMENT,
        username TEXT UNIQUE, password TEXT, email TEXT UNIQUE, firstName TEXT,
        lastName TEXT, address TEXT, phone TEXT, color TEXT)");
    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
        db.execSQL("DROP TABLE IF EXISTS " +TABLE_NAME); //Drop older table if exists
        onCreate(db);
    }
}
```

Παράρτημα Α2 : Έλεγχος και εισαγωγή στοιχείων Χρήστη στη Βάση Δεδομένων

```
//Checking if the username is already used
Cursor cursorusername = db.rawQuery("SELECT * FROM " + DatabaseHelper.TABLE_NAME +
    " WHERE " + DatabaseHelper.COL_2 + "=?",new String[]{username});
if (cursorusername != null)
{
    if(cursorusername.getCount()>0)
    {
        cursorusername.moveToNext();
        check1 = false;
    }else
    {
        check = true;
    }
}

//Checking if the email is already used
Cursor cursoremail = db.rawQuery("SELECT * FROM " + DatabaseHelper.TABLE_NAME +
    " WHERE " + DatabaseHelper.COL_4 + "=?",new String[]{email});
if (cursoremail != null)
{
    if(cursoremail.getCount()>0)
    {
        cursoremail.moveToNext();
        check2 = false;
    }
    else
    {
        check2 = true;
    }
}

//If both username and email are unique then insert data in database and clear the fields
db=openHelper.getWritableDatabase();
insertdata(firstName,lastName,email,address,phone,username,password,cchun);
db.close();

//insertData helps us to add our data to the database
public void insertdata(String firstName, String lastName,
    String email, String address, String phone, String username, String password, String cchun){
    ContentValues contentValues = new ContentValues();
    contentValues.put(DatabaseHelper.COL_2, username);
    contentValues.put(DatabaseHelper.COL_3, password);
    contentValues.put(DatabaseHelper.COL_4, email);
    contentValues.put(DatabaseHelper.COL_5, firstName);
    contentValues.put(DatabaseHelper.COL_6, lastName);
    contentValues.put(DatabaseHelper.COL_7, address);
    contentValues.put(DatabaseHelper.COL_8, phone);
    contentValues.put(DatabaseHelper.COL_9, cchun);
    long id = db.insert(TABLE_NAME,null, contentValues);
}
```

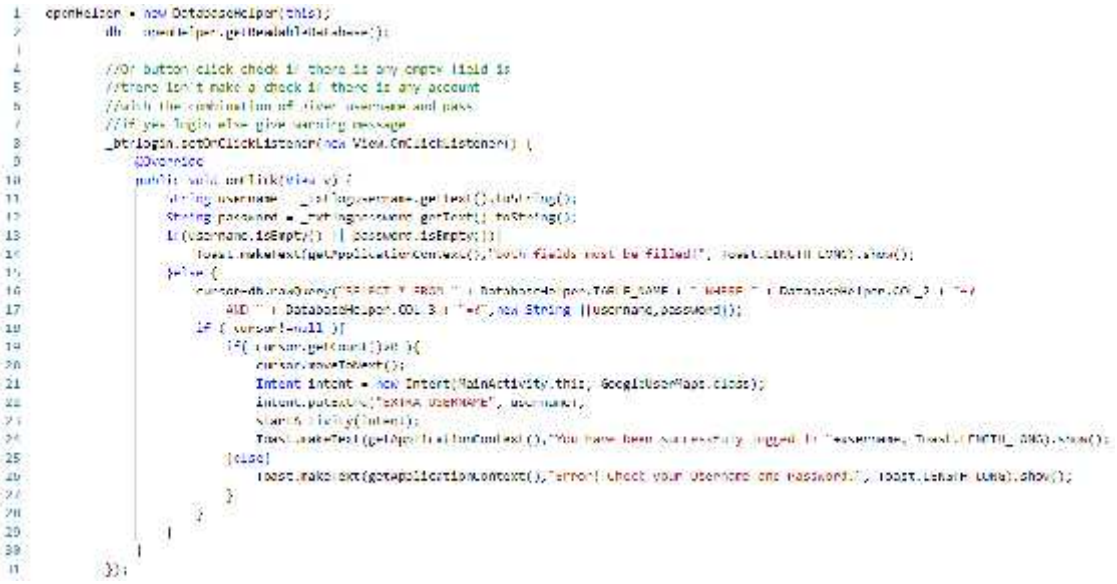
Παράρτημα Α3 : Μεταφορά από ένα Activity σε ένα άλλο

```
//on button click open register activity
_btnmainregister.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Intent intent = new Intent(MainActivity.this, RegisterActivity.class);
        startActivity(intent);
    }
});
```

Παράρτημα Α4 : Ολοκλήρωση Register Activity

```
//when cancel button is clicked then finish register activity and return to main
_btncancel.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        RegisterActivity.this.finish();
    }
});
```

Παράρτημα Α5 : Έλεγχος Εγκυρότητας Στοιχείων Χρήστη



Παράρτημα Α6 : Δημιουργία Χάρτη και Εύρεση τοποθεσίας Χρήστη



```
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="com.google.android.providers.gsf.permission.READ_GSERVICES" />
```

```
public void onMapReady(GoogleMap googleMap)
{
    mMap = googleMap;

    if (ContextCompat.checkSelfPermission(this, Manifest.permission.ACCESS_FINE_LOCATION)== PackageManager.PERMISSION_GRANTED)
    {
        buildGoogleApiClient();
        mMap.setMyLocationEnabled(true);
    }

}

public boolean checkAndRequestPermissions()
{
    if (ContextCompat.checkSelfPermission(this, Manifest.permission.ACCESS_FINE_LOCATION)   PackageManager.PERMISSION_GRANTED) {
        if(ActivityCompat.shouldShowRequestPermissionRationale(this,Manifest.permission.ACCESS_FINE_LOCATION))
        {
            ActivityCompat.requestPermissions(this,new String []{Manifest.permission.ACCESS_FINE_LOCATION}, Request_User_Location_Code);
        }
        else
            {
            ActivityCompat.requestPermissions(this,new String[]{Manifest.permission.ACCESS_FINE_LOCATION}, Request_User_Location_Code);
        }
        return false;
    }
    else
    {
        return true;
    }
}


@Override
public void onLocationChanged(Location location)
{
    latitude = location.getLatitude();
    longitude = location.getLongitude();

    lastLocation = location;

    if(currentUserLocationMarker != null)
    {
        currentUserLocationMarker.remove();                                                                    ANTED)
    }

    LatLng latLng = new LatLng(location.getLatitude(),location.getLongitude());
    MarkerOptions markerOptions = new MarkerOptions();
    markerOptions.position(latLng);
    markerOptions.title("User's Current Location");
    markerOptions.icon(BitmapDescriptorFactory.defaultMarker(BitmapDescriptorFactory.HUE_RED));

    currentUserLocationMarker = mMap.addMarker(markerOptions);
    mMap.moveCamera(CameraUpdateFactory.newLatLngZoom(latLng,12));

    if(googleApiClient != null)
    {
        LocationServices.FusedLocationApi.removeLocationUpdates(googleApiClient,this);
    }

}

@Override
public void onConnected(@Nullable Bundle bundle)
{
    locationRequest=new LocationRequest();
    locationRequest.setInterval(1100);
    locationRequest.setFastestInterval(1100);
    locationRequest.setPriority(locationRequest.PRIORITY_BALANCED_POWER_ACCURACY);

    if (ContextCompat.checkSelfPermission(this, Manifest.permission.ACCESS_FINE_LOCATION)== PackageManager.PERMISSION_GRANTED)
    {
        LocationServices.FusedLocationApi.requestLocationUpdates(googleApiClient, locationRequest, this);
    }

}
```

Παράρτημα Α7 : Βήματα αναζήτησης πλησιέστερων τοποθεσιών

```
String gasStation = "gas_station", market = "supermarket", cinema = "movie_theater";
        Status url;
        Object transferData[] = new Object[6];
        GetNearbyPlaces getNearbyPlaces = new GetNearbyPlaces(this);

        //Get User Location
        LatLng latLngUser = new LatLng(latitude, longitude);
        MarkerOptions userLocMarkerOptions = new MarkerOptions();
        userLocMarkerOptions.position(latLngUser);
        userLocMarkerOptions.title("User's Current Location");
        userLocMarkerOptions.icon(BitmapDescriptorFactory.defaultMarker(BitmapDescriptorFactory.HUE_ORANGE));

        ΠΑΡΑΚΟΛΟΥΘΗΣΗ ΤΟΠΟΘΕΣΙΑΣ ΠΑΡΑΚΑΤΩ ΑΠΟ ΑΝΑΖΗΤΗΣΗ
        case R.id.gasStation_nearby:
                mMap.clear();
                mMap.addMarker(userLocMarkerOptions);

                url = getUrl(latitude,longitude,gasStation);
                transferData[0] = mMap;
                transferData[1] = url;
                transferData[2] = latitude;
                transferData[3] = longitude;
                transferData[4] = "Gas Station";
                transferData[5] = username;

                getNearbyPlaces.execute(transferData);

                Toast.makeText(this, "Searching for the nearest Gas Station.", Toast.LENGTH_SHORT).show();
                break;
                private String getUrl(double latitude, double longitude, String nearbyPlace) {
                    StringBuilder googleURL = new StringBuilder("https://maps.googleapis.com/maps/api/place/nearbysearch/json?");
                    googleURL.append("location=" + latitude + "," + longitude);
                    googleURL.append("&radius=" + ProximityRadius);
                    googleURL.append("&type=" + nearbyPlace);
                    googleURL.append("&sensor=true");
                    googleURL.append("&key=" + "AIzaSyBAhBlUF-bnnbJQIaaTc4JMR1-DQkUET");

                    Log.d("GoogleUserMaps", "Url = " + googleURL.toString());

                    return googleURL.toString();
                }
```

Παράρτημα Α8 : Αναζήτηση κοντινότερου ανοικτού προορισμού

```
protected void onPostExecute(String s) {
    DataParser dataParser = new DataParser();

    //Parse the data to get all nearby places
    List < HashMap < String, String >> nearbyPlacesList = null;
    nearbyPlacesList = dataParser.parse(s);

    //Get data for all nearby places and choose the best shortest one
    //<---------- START OF "IF" WHILE LOOP ---------->
    while (loopCheck == true) {
     for (int i = 0; i < nearbyPlacesList.size(); i++) {

      MarkerOptions markerOptions = new MarkerOptions();
      HashMap < String, String > googleNearbyPlace = nearbyPlacesList.get(i);

      String nameOfPlace = googleNearbyPlace.get("place_name");
      String vicinity = googleNearbyPlace.get("vicinity");
      String reference = googleNearbyPlace.get("reference");
      Log.d("Info", "name -> " + nameOfPlace + " vicinity -> " + vicinity + " reference -> " + reference);

      if (refList.isEmpty()) {
       refList.add(reference);
      } else if (!refList.contains(reference)) {
       refList.add(reference);
      }

      double lat = Double.parseDouble(googleNearbyPlace.get("lat"));
      double lng = Double.parseDouble(googleNearbyPlace.get("lng"));

      LatLng latLng = new LatLng(lat, lng);

      //Get the JSON url from GoogleMaps
      url2 = getDirectionsUrl(lat, lng);
      googleDirectionData = downloadDirectionsUrl(url2);

      //Parse the data to get the Duration and the Distance from JSON
      HashMap < String, String > directionsList = null;
      directionsList = dataParser.parseDirections(googleDirectionData);

      //Get the duration and distance
      duration = directionsList.get("duration");
      distance = directionsList.get("distance");

      //Get the duration and distance without characters
      der = Integer.parseInt(duration.replaceAll("\\D+", ""));
      dist = Integer.parseInt(distance.replaceAll("\\D+", ""));

      //Check if there are places without Date details
      ////////////////////////////////////////////////////////////////////////////////////////////////////
      if (noDate.isEmpty()) {
       Log.d("EINAI ADIA", "EINAI ADIA");
       listCheck = true;
      } else if (!noDate.contains(reference)) {
       Log.d("DEN IPARXEI", "DEN IPARXEI");
       listCheck = true;
      } else if (noDate.contains(reference)) {
       Log.d("IPARXEI", "IPARXEI");
       listCheck = false;
      }
      ////////////////////////////////////////////////////////////////////////////////////////////////////

      //Check if a place is on heartbeat ll..
      ////////////////////////////////////////////////////////////////////////////////////////////////////
      if (heartBeat.isEmpty()) {
       Log.d("EINAI ADIA", "EINAI ADIA");
       listCheck2 = true;
      } else if (!heartBeat.contains(reference)) {
       Log.d("DEN IPARXEI", "DEN IPARXEI");
       listCheck2 = true;
      } else if (heartBeat.contains(reference)) {
       Log.d("IPARXEI", "IPARXEI");
       listCheck2 = false;
      }
      ////////////////////////////////////////////////////////////////////////////////////////////////////
```



```
154
155     //FIND EXACT TIME OF PHONE AND ADD THE DURATION TIME TO GET TO THE WILLING POINT
156     Calendar calendar = Calendar.getInstance(Locale.getDefault());
157     year = calendar.get(Calendar.YEAR);
158     month = calendar.get(Calendar.MONTH)
159     day = calendar.get(Calendar.DAY_OF_MONTH);
160
161     int hour = calendar.get(Calendar.HOUR_OF_DAY);
162     int minute = calendar.get(Calendar.MINUTE);
163
164
165     String pT = day + "/" + (month + 1) + "/" + year + "," + hour + ":" + minute;
166     String datesqlCheck = day + "/" + (month + 1) + "/" + year;
167
168
169     Log.d("Hour", "" + pT);
170     SimpleDateFormat df = new SimpleDateFormat("dd/MM/yyyy,HH:mm");
171     SimpleDateFormat dfarrtime = new SimpleDateFormat("HH:mm");
172     Date d2 = null;
173     Date pre0 = null;
174     try {
175      d2 = df.parse(pT);
176      pre0 = df.parse(pT);
177     } catch (ParseException e) {
178      e.printStackTrace();
179     }
180     calendar.setTime(d2);
181
182     calendar.add(Calendar.MINUTE, minDur);
183
184     String arrivalTime = dfarrtime.format(calendar.getTime());
185
186     //If user search for market +40minutes extra
187     if (searchFor.equals("Market")) {
188      calendar.add(Calendar.MINUTE, 40);
189     }
190
191     pT = df.format(calendar.getTime());
192
193     try {
194      d2 = df.parse(pT);
195     } catch (ParseException e) {
196      e.printStackTrace();
197     }
198     ////////////////////////////////////////////////////////////////////////////////////////////////
199
200     Log.d("WHAT SEARCHED + DATE", "SEARCHED FOR -> " + searchFor + " WITH REF-> " + ref + " | DATE AND TIME -> " + dayNam + "  | " + d2);
201     Log.d("TEST", "" + hourList[0]);
202
203     //Set up DatabaseHelper
204     openHelper = new DatabaseHelper(mContext);
205
206     //Find Place's Open/Close Date and Time
207     //START OF CHECKING HOURS
208     //1ST CONDITION IF 24/7
209     if (hourList[0].equals("0.0000")) {
210      //Parse and show shortest path to destination
211
212      //Checking if user has an appointment in the same time
213      db = openHelper.getReadableDatabase();
214      cursor = db.rawQuery("SELECT username, appNickname, dateSelected, startTime, endTime FROM " + DatabaseHelper.TABLE_NAME2 + " WHERE "
215        + DatabaseHelper.COL_11 + "=? AND " + DatabaseHelper.COL_13 + "=?", new String[] {
216       username,
217       datesqlCheck
218      });
219      Boolean validDateChecker = true;
220      if (cursor.moveToFirst()) {
221       while (!cursor.isAfterLast()) {
222        dbAppnickname = cursor.getString(cursor.getColumnIndex("appNickname"));
223        String dbDateSelected = cursor.getString(cursor.getColumnIndex("dateSelected"));
224        String dbStartTime = cursor.getString(cursor.getColumnIndex("startTime"));
225        String dbEndTime = cursor.getString(cursor.getColumnIndex("endTime"));
226
227        Log.d("dbDateSelected", "" + dbDateSelected);
228        Log.d("dbStartTime", "" + dbStartTime);
229        Log.d("dbEndTime", "" + dbEndTime);
230
231        SimpleDateFormat sdfsqltimes = new SimpleDateFormat("dd/MM/yyyy,HH:mm");
232        Date sqlstartTime = null;
233        Date sqlendTime = null;
234        try {
235         sqlstartTime = sdfsqltimes.parse(dbDateSelected + "," + dbStartTime);
236         sqlendTime = sdfsq.times.parse(dbDateSelected + "," + dbEndTime);
237        } catch (ParseException e) {
238         e.printStackTrace();
239        }
240
```

```
```





```
    //Function that gets the url from googleMaps and convert it to String
    private String getDirectionsUrl(Double end_latitude, Double end_longitude) {
     StringBuilder googleDirectionsUrl = new StringBuilder("https://maps.googleapis.com/maps/api/directions/json?");
     googleDirectionsUrl.append("origin=" + latitude + "," + longitude);
     googleDirectionsUrl.append("&destination=" + end_latitude + "," + end_longitude);
     googleDirectionsUrl.append("&key=" + "AIzaSyBAbb8LvE-orobi2qJaaJc92MB1-DQkNoI");

     return googleDirectionsUrl.toString();
    }

    //Function that download the URL
    public String downloadDirectionsUrl(String url) {
     DownloadUrl downloadUrl = new DownloadUrl();
     try {
      googleData = downloadUrl.ReadTheURL(url);
     } catch (IOException e) {
      e.printStackTrace();
     }

     return googleData;
    }


    //Function which work is to display the shortest path
    public void displayDirection(String[] directionsList) {
     int count = directionsList.length;
     for (int i = 0; i < count; i++) {
      PolylineOptions options = new PolylineOptions();
      options.color(Color.RED);
      options.width(15);
      options.addAll(PolyUtil.decode(directionsList[i]));

      mMap.addPolyline(options);
     }
    }
```

```
public class DownloadUrl
{
    public String ReadTheURL(String placeURL) throws IOException {
        String Data = "";
        InputStream inputStream = null;
        HttpURLConnection httpURLConnection = null;
        //Log.d("Request","sencing");
        try {
            URL url = new URL(placeURL);
            httpURLConnection = (HttpURLConnection) url.openConnection();
            httpURLConnection.connect();

            inputStream = httpURLConnection.getInputStream();
            BufferedReader bufferedReader = new BufferedReader(new InputStreamReader(inputStream));
            StringBuffer stringBuffer = new StringBuffer();

            String line = "";

            while ((line = bufferedReader.readLine()) != null) {
                stringBuffer.append(line);
            }

            Data = stringBuffer.toString();
            bufferedReader.close();
        } catch (MalformedURLException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        } finally {
            if(inputStream!=null)
            inputStream.close();
            httpURLConnection.disconnect();
        }

        return Data;
    }
}
```

```
public class DataParser
{
    //SPLIT JSON FOR DURATION AND DISTANCE
    private HashMap<String,String> getDuration(JSONArray googleDirectionsJson)
    {
        HashMap<String,String> googleDirectionsMap = new HashMap<>();
        String duration = "";
        String distance = "";

        //Log.d("JSON RESPONSE", googleDirectionsJson.toString());
        try
        {
            duration = googleDirectionsJson.getJSONObject(0).getJSONObject("duration").getString("text");
            distance = googleDirectionsJson.getJSONObject(0).getJSONObject("distance").getString("text");

            googleDirectionsMap.put("duration", duration);
            googleDirectionsMap.put("distance", distance);

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

        return googleDirectionsMap;
    }


    //SPLIT JSON FOR NEARBY PLACES
    private HashMap<String, String> getSingleNearbyPlace(JSONObject googlePlaceJSON )
    {
        HashMap<String, String> googlePlaceMap = new HashMap<>();
        String NameOfPlace = "-NA-";
        String vicinity = "-NA-";
        String latitude = "";
        String longitude = "";
        String reference = "";

        try
        {
            if(!googlePlaceJSON.isNull("name"))
            {
                NameOfPlace = googlePlaceJSON.getString("name");
            }

            if(!googlePlaceJSON.isNull("vicinity"))
            {
                vicinity = googlePlaceJSON.getString("vicinity");
            }

            latitude = googlePlaceJSON.getJSONObject("geometry").getJSONObject("location").getString("lat");
            longitude = googlePlaceJSON.getJSONObject("geometry").getJSONObject("location").getString("lng");
            reference = googlePlaceJSON.getString("reference");

            googlePlaceMap.put("place_name", NameOfPlace);
            googlePlaceMap.put("vicinity", vicinity);
            googlePlaceMap.put("lat", latitude);
            googlePlaceMap.put("lng", longitude);
            googlePlaceMap.put("reference", reference);

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

        return googlePlaceMap;
    }

```

```
    private List<HashMap<String,String>> getAllNearbyPlaces(JSONArray jsonArray)
    {
        int counter = jsonArray.length();

        List<HashMap<String,String>> NearbyPlaceslist = new ArrayList<>();

        HashMap<String,String> NearbyplaceMap = null;

        for(int i=0; i<counter; i++)
        {
            try
            {
                NearbyplaceMap = getSingleNearbyPlace( (JSONObject) jsonArray.get(i) );
                NearbyPlaces.list.add(nearbyplacemap);
            }
            catch (JSONException e)
            {
                e.printStackTrace();
            }
        }
        return NearbyPlacesList;
    }

    public List<HashMap<String, String>> parse(String jSONdata)
    {
        JSONArray jsonArray = null;
        JSONObject jsonObject;

        try
        {
            jsonObject = new JSONObject(jSONdata);
            jsonArray = jsonObject.getJSONArray("results");
        }
        catch (JSONException e)
        {
            e.printStackTrace();
        }

        return getAllNearbyPlaces(jsonArray);
    }

    public  HashMap<String, String> parseDuration(String jsondata)
    {
        JSONArray jsonArray = null;
        JSONObject jsonObject;

        try
        {
            jsonObject = new JSONObject(jsondata);
            jsonArray = jsonObject.getJSONArray("routes").getJSONObject(0).getJSONArray("legs");
        }
        catch (JSONException e)
        {
            e.printStackTrace();
        }

        return getDuration(jsonArray);


    }


    //Split the JSON for closing hours
    public  String[] parseHours(String jSONdata)
    {
        JSONArray jsonArray = null;
        JSONObject jsonObject;

        try
        {
            jsonObject = new JSONObject(jsondata);
            if( jsonObject.getJSONObject("result").has("opening_hours") )
            {
                jsonArray = jsonObject.getJSONObject("result").getJSONObject("opening_hours").getJSONArray("periods");
            }else
            {
                return null;
            }


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

        return getHours(jsonArray);
    }
```

```
    public String[] getHours(JSONArray googleHoursJson)
    {
        int count = googleHoursJson.length();
        String[] hours = new String[count];

        for(int i=0; i<count; i++)
        {
            try
            {
                hours[i] = getHour(googleHoursJson.getJSONObject(i));
            }
            catch (JSONException e)
            {
                e.printStackTrace();
            }
        }

        return hours;
    }

    public String getHour(JSONObject googleHourJson)
    {
        StringBuilder hour = new StringBuilder();

        try
        {
            if ( googleHourJson.has("open") )
            {
                hour.append(googleHourJson.getJSONObject("open").getString("day"));
                hour.append(",");
                hour.append(googleHourJson.getJSONObject("open").getString("time"));
            }

            if ( ( googleHourJson.has("open") ) && ( googleHourJson.has("close") ) )
            {
                hour.append(",");
            }

            if ( googleHourJson.has("close") )
            {
                hour.append(googleHourJson.getJSONObject("close").getString("day"));
                hour.append(",");
                hour.append(googleHourJson.getJSONObject("close").getString("time"));
            }
        }

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

        return hour.toString();
    }

    //SPLIT JSON FOR PATHS
    public String[] parseDirections(String jSONdata)
    {
        JSONArray jsonArray = null;
        JSONObject jsonObject;

        try
        {
            jsonObject = new JSONObject(jSONdata);
            jsonArray = jsonObject.getJSONArray("routes").getJSONObject(0).getJSONArray("legs").getJSONObject(0).getJSONArray("steps");
        }
        catch (JSONException e)
        {
            e.printStackTrace();
        }
        return getPaths(jsonArray);
    }

    public String[] getPaths(JSONArray googleStepsJson)
    {
        int count = googleStepsJson.length();
        String[] polylines = new String[count];

        for(int i=0; i<count; i++)
        {
            try
            {
                polylines[i] = getPath(googleStepsJson.getJSONObject(i));
            } catch (JSONException e)
            {
                e.printStackTrace();
            }
        }
        return polylines;
    }
```

```
251     public String getPath(JSONObject googlePathJson)
252     {
253         String polyline = "";
254 
255         try
256         {
257             polyline = googlePathJson.getJSONObject("polyline").getString("points");
258         }
259         catch (JSONException e)
260         {
261             e.printStackTrace();
262         }
263 
264         return polyline;
265     }
266 }
```