ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ – ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Πρόγραμμα Μεταπτυχιακών Σπουδών

«Προηγμένα Συστήματα Πληροφορικής – Ανάπτυξη Λογισμικού και Τεχνητής Νοημοσύνης»

Μεταπτυχιακή Διατριβή

Τίτλος Διατριβής	**Συνδυασμός Android και Tensorflow με στόχο την παραγωγή analytics για recommendations για τους χρήστες, βάσει συγκεκριμένων παρατηρήσεων/δεδομένων που συλλέγει για αυτούς η κάθε συσκευή.** **Combination of Android and Tensorflow to produce analytics for recommendations for users, based on specific observations/data that collected about them by each device.**
Ονοματεπώνυμο Φοιτητή	**Ανδρικόπουλος Βασίλειος**
Πατρώνυμο	**Κωνσταντίνος**
Αριθμός Μητρώου	**ΜΠΣΠ21004**
Επιβλέπων	**Αλέπης Ευθύμιος, Καθηγητής**

Ημερομηνία Παράδοσης, **Δεκέμβριος 2024**

Τριμελής Εξεταστική Επιτροπή

Ευθύμιος Αλέπης Καθηγητής	Μαρία Βίρβου Καθηγήτρια	Ευάγγελος Σακκόπουλος Αναπληρωτής Καθηγητής

1. Περίληψη

Στο πλαίσιο της παρούσας διπλωματικής εργασίας πραγματοποιήθηκε η υλοποίηση μιας εφαρμογής Android, στόχος της οποίας είναι να καταγράφει δεδομένα από τη συσκευή ενός χρήστη, κατά την κίνησή του, καθ' όσο αυτός πραγματοποιεί μια διαδρομή περπατώντας (κίνηση με τα πόδια, όχι με κάποιο μεταφορικό μέσο). Στόχος της καταγραφής δεδομένων από τη συσκευή του χρήστη είναι η ανάλυση και επεξεργασία τους, ώστε με τη βοήθεια κατάλληλων τεχνικών μηχανικής μάθησης που υλοποιήθηκαν, με χρήση Tensorflow, να γίνει η εξαγωγή κάποιων συμπερασμάτων ενδιαφέροντος. Πιο συγκεκριμένα, στην εν λόγω εργασία, μας ενδιέφερε να εξετάσουμε εάν θα μπορούσε να βγει κάποιο συμπέρασμα σχετικά με τον αριθμό των βημάτων που θα μπορούσε να περπατήσει ένας χρήστης, στην περιοχή της Αθήνας, βασισμένο σε παράγοντες, όπως ο καιρός και η γεωγραφική θέση του χρήστη, σε συνδυασμό με τον χρόνο. Επόμενος στόχος ήταν, αυτά τα συμπεράσματα να παρουσιάζονται στον χρήστη της εφαρμογής ως προτάσεις, όταν ξεκινάει την διαδρομή του. Με αυτόν τον τρόπο δίνεται μια πρόταση στο χρήστη ανάλογα με την περιοχή που βρίσκεται, σε συνδυασμό με τις καιρικές συνθήκες που επικρατούν, για τον μέσο όρο βημάτων που συνίσταται να διανύσει.

1. Abstract

In the context of this thesis, the implementation of an Android application was carried out, the aim of which is to record data from a user's device, during his/her movement, while he/she is walking (movement on foot, not with any means of transport). The aim of recording data from the user's device is to analyze and process them, so that with the help of appropriate machine learning techniques implemented, using Tensorflow, some conclusions of interest can be drawn. More specifically, in this work we were interested in examining whether some inference could be drawn about the number of steps a user could walk, in the Athens area, based on factors such as the weather and the geographical location of the user, combined with time. The next goal was to present these conclusions to the user of the application as suggestions when he/she starts his/her route. In this way a suggestion is given to the user depending on the area he is in, combined with the weather conditions, for the average steps they are advised to take.

2. Περιεχόμενα

3. Ανασκόπηση Πεδίου

Καθημερινά οι άνθρωποι χρειάζονται να περπατήσουν κάποιον αριθμό βημάτων είτε για να μεταβούν στο χώρο εργασίας τους, είτε για να πραγματοποιήσουν δραστηριότητες/εξωτερικές υποθέσεις, είτε πηγαίνοντας βόλτα, είτε για την αύξηση/βελτίωση της φυσικής τους δραστηριότητας. Συνολικά, τα τελευταία χρόνια βάσει των ρυθμών εργασίας, των εξελίξεων που έφερε ο ιός του Covid-19, όπως η εξ ’αποστάσεως εργασία, το πλήθους κόσμου που ζει σε συνοικίες μέσα στον αστικό ιστό, με περιορισμένο πλήθος χώρων για φυσική δραστηριότητα, έχει δημιουργηθεί η ανάγκη αύξησης της φυσικής δραστηριότητας των ανθρώπων, κυρίως για λόγους υγείας. Κάθε άνθρωπος φυσικά διαθέτει την δική του φυσική κατάσταση, αντοχή, προθυμία και το χρόνο για να διανύσει έναν ορισμένο αριθμό βημάτων.

Η φυσική δραστηριότητα όπως υπολογίζεται από τον αριθμό βημάτων που διανύουν οι άνθρωποι καθημερινά, αποτελεί έναν από τους σημαντικούς δείκτες για την υγεία τους. Ως ιδανικός στόχος για τη διατήρηση της υγείας των ανθρώπων, παραδοσιακά, είχε προωθηθεί ο αριθμός των 10.000 βημάτων ημερησίως. Ωστόσο, πιο σύγχρονες μελέτες, έδειξαν ότι ακόμη και μικρότερος αριθμός βημάτων ημερησίως, μπορεί να προσφέρει σημαντικά οφέλη.

Σύμφωνα με μελέτη που δημοσιεύτηκε στο British Journal of Sports and Medicine, κάθε επιπλέον βήμα πέραν των 2.200 ημερησίως μειώνει τον κίνδυνο της καρδιακής νόσου και πρόωρου θανάτου. Ο χαμηλότερος κίνδυνος παρατηρήθηκε σε άτομα που έκαναν από 9.000 έως 10.500 βήματα ημερησίως. Μια ακόμα μελέτη με τίτλο “Association of Step Volume and Intensity With All-cause Mortality in Older Women” δείχνει ότι τα 4.400 βήματα ημερησίως συνδέονται με μείωση της θνησιμότητας, ενώ τα οφέλη αυξάνονται έως τα 7.500 βήματα. Η τελευταία επίσης, υπογραμμίζει ότι τα σύγχρονα τεχνολογικά μέσα όπως οι φορητές συσκευές τηλεφώνου μπορούν να ενθαρρύνουν την αύξηση της καθημερινής κίνησης, σε συνδυασμό με την ενσωμάτωση και παρουσίαση συστάσεων βασισμένων στον αριθμό των βημάτων.

Συνεπώς, η αύξηση της φυσικής δραστηριότητας, ως επακολουθώ της αύξησης των ημερήσιων βημάτων που κάνει ένας άνθρωπος είναι αναγκαία για τα οφέλη που μπορεί να προσφέρει στην υγεία του. Βέβαια, πολλές φορές η κίνηση λόγω διαφόρων παραγόντων μπορεί να μην είναι επιθυμητή ή ασφαλής, όπως για παράδειγμα, όταν επικρατούν σε μια περιοχή έντονα καιρικά φαινόμενα (δυνατή βροχή, χαλάζι, χιόνι, σκόνη στην ατμόσφαιρα, κτλ), όταν η θερμοκρασία είναι πολύ υψηλή ή πολύ χαμηλή, ή σε περιοχές που έχουν συνωστισμένη κίνηση οχημάτων, αυτοκινητοδρόμους ή περιοχές που είναι δύσβατες και απομονωμένες.

Η εν λόγω διπλωματική εργασία, θέλει να συμβάλει σε αυτόν τον τομέα, προτείνοντας στους χρήστες της, έναν μέσο όρο βημάτων που τους συνίσταται να πραγματοποιήσουν δεδομένης της γεωγραφικής τους τοποθεσίας, δεδομένων των καιρικών συνθηκών και της θερμοκρασίας τη στιγμή που ξεκινούν μια διαδρομή, καθώς και της ώρας εντός της ημέρας και της ημέρας της εβδομάδας.

4. Τεχνολογίες Υλοποίησης

4.1 Android – Kotlin

Το Android είναι ένα λειτουργικό σύστημα για φορητές συσκευές με οθόνη αφής κυρίως, όπως είναι τα smartphones και τα tablet. Επιπλέον, υποστηρίζει μια μεγάλη γκάμα συσκευών από smartwatches, smart TVs (Android TV) οθόνες αυτοκινήτων (Android Auto) και άλλες IoT συσκευές. Είναι λειτουργικό σύστημα που έχει αναπτύξει η Google, βασισμένο σε μια τροποποιημένη έκδοση του πυρήνα του Linux και είναι χαρακτηρισμένο ως λογισμικό ανοιχτού κώδικα (open source). Η Google παρέχει τον πηγαίο κώδικα του Android μέσω του Android Open Source Project (AOSP). Αυτήν τη στιγμή το Android κατέχει το 45% της παγκόσμιας αγοράς λειτουργικών συστημάτων ακολουθούμενο από τα Windows που κατέχουν το 26%.

Το Android διαθέτει 15 εκδόσεις με το Android 15 να είναι η πιο πρόσφατη έκδοσή του που κυκλοφόρησε στις 3 Σεπτεμβρίου του 2024. Διαθέτει μια ευέλικτη και εύκολα τροποποιήσιμη διεπαφή χρήστη (UI), παρέχοντας στους χρήστες πολλές δυνατότητες εξατομίκευσης. Η Google έχει ενσωματώσει στο Android μια σειρά από εφαρμογές που ανήκουν στη σουίτα της και μερικές από αυτές που περιλαμβάνουν είναι οι χάρτες, το ημερολόγιο και το Gmail. Επιπρόσθετος, το Play Store που διαθέτει το σύστημα του Android παρέχει στους χρήστες τη δυνατότητα πρόσβασης σε μια ευρεία γκάμα εφαρμογών ψυχαγωγίας, εκπαίδευσης, εφαρμογών-εργαλείων και άλλων τύπων. Οι εφαρμογές που υποστηρίζονται από το Android υλοποιούνται από προγραμματιστές με χρήση των γλωσσών Java, Kotlin και C++.

Για την υλοποίηση εφαρμογών Android, η Google σε συνεργασία με την JetBrains έχουν δημιουργήσει το Android Studio, το οποίο είσαι βασισμένο στο IntelliJ IDEA IDE. Το Android Studio αποτελεί το επίσημο ολοκληρωμένο περιβάλλον (IDE) ανάπτυξης εφαρμογών Android. Παρέχει όλα τα απαραίτητα εργαλεία για την κατασκευή διεπαφών χρήστη (UI) τόσο με χρήση drag ‘n drop γραφικών στοιχείων, όσο και με συγγραφή κώδικα xml. Επιπλέον, παρέχει τη δυνατότητα δημιουργίας εικονικών συσκευών μέσω εξομοιωτή (emulator) για την δοκιμή εφαρμογών, ώστε να μην χρειάζεται φυσική συσκευή, απευθείας στη διάθεση του προγραμματιστή. Διαθέτει εργαλεία δοκιμής (testing) και εκσφαλμάτωσης (debugging), ώστε οι προγραμματιστές να μπορούν να εντοπίζουν και να διορθώνουν λάθη για την βελτίωση της απόδοσης και της ποιότητας της εφαρμογής.Μια ακόμη εφαρμογή είναι η εφαρμογή του Android Studio που ενσωματώνει εργαλεία ελέγχου έκδοσης, όπως το Git, για την ευκολότερη αποθήκευση και ανανέωση του κώδικα σε αποθετήρια (repositories).

Όπως αναφέρθηκε και νωρίτερα, εντός του συνόλου γλωσσών που υποστηρίζονται από το Android για την ανάπτυξη εφαρμογών, είναι η Kotlin. Η Kotlin είναι μια διαπλατφορμική (cross-platform), στατικού τύπου, γενικής χρήσης, αντικειμενοστραφής γλώσσα προγραμματισμού με παρεμβολή τύπου (type inference), που έχει δημιουργήσει η JetBrains. Η Kotlin είναι διαλειτουργική με την εικονική μηχανή της Java (JVM), τις βιβλιοθήκες κλάσεων της Java και το Android. Η εν λόγω γλώσσα αρχικά σχεδιάστηκε, για να βελτιώσει τη λειτουργία της Java και χρησιμοποιείται συχνά σε συνδυασμό με την Java. Η Kotlin έχει οριστεί πλέον από την Google, ως η προτεινόμενη γλώσσα για την ανάπτυξη εφαρμογών Android και παρά το γεγονός αυτό, η διαλειτουργηκότητά της με την Java έχει οδηγήσει στη χρήση της και σε πολλούς άλλους τύπους εφαρμογών (backend, frontend, full-stack, κ.α.).

Τα πλεονεκτήματα που προσφέρει η Kotlin σε σύγκριση με την Java είναι:

- Απλότητα και καθαρότητα στον κώδικα, μιας και ο κώδικας της Kotlin είναι πιο σύντομος και καθαρός, προσφέροντας την ίδια λειτουργικότητα
- Προσφέρει ασφάλεια για τον τύπο null εξαλείφοντας τα null NullPointerException σφάλματα

- Όπως αναφέρθηκε και νωρίτερα η διαλειτουργικότητα της με την Java, διευκολύνει την μετάβασή της από την Java μιας και υποστηρίζει τις ίδιες βιβλιοθήκες
- Διαθέτει μεγάλη υποστήριξη από την Google με εργαλεία και βιβλιοθήκες, ώστε να είναι βελτιωμένα για την χρήση της
- Αυξανόμενη υποστήριξη από το προγραμματιστικό κοινό

4.1.1 Kotlin Coroutines

Το Kotlin Coroutines είναι ένα χαρακτηριστικό της Kotlin που υποστηρίζεται στο Android με την εγκατάσταση της ομώνυμης βιβλιοθήκης και προσφέρει την δυνατότητα ανάπτυξης μπλοκ ασύγχρονου κώδικα στις εφαρμογές. Η λειτουργικότητα που προσφέρει η Kotlin μέσω αυτής της βιβλιοθήκης είναι πολύ σημαντική, βάση των σύγχρονων προγραμματιστικών προτύπων μιας και διευκολύνει την υλοποίηση μίας μεγάλης γκάμας από λειτουργίες που απαιτούν πόρους ή δεδομένα οι οποίες μπορούν να εκτελούνται στο παρασκήνιο, χωρίς να μπλοκάρουν σημαντικά μέρη της εφαρμογής, όπως το UI. Μια τέτοια περίπτωση λειτουργίας, είναι η αναμονή δεδομένων από ένα API, χωρίς να μπλοκάρεται η ροή του υπόλοιπου κώδικα.

4.1.2 Dagger - Hilt

Το Dagger Hilt είναι μια βιβλιοθήκη για την υλοποίηση dependency injection (DI) στο Android, που βασίζεται στο Dagger 2. Το Hilt έχει σχεδιαστεί για να απλοποιεί την υλοποίηση του dependency injection σε εφαρμογές Android, μειώνοντας τον boilerplate κώδικα και παρέχοντας απρόσκοπτη ενσωμάτωση με στοιχεία του Android, όπως Activities, Fragments, Services και ViewModels. Αξιοποιεί την ισχύ του Dagger, ενώ προσφέρει ένα πιο συνοπτικό και διαισθητικό API για τους προγραμματιστές, με χρήση Annotations. Λειτουργεί σε χρόνο μεταγλώττισης, οπότε η λήψη σφαλμάτων γίνεται σε χρόνο μεταγλώττισης, αντί για χρόνο εκτέλεσης, κάτι που είναι ασφαλέστερο σε σύγκριση με άλλες μεθόδους.

4.1.3 Retrofit - OkHttp

Το Retrofit είναι ένας ασφαλής ως προς τον τύπο (type safe) Http client υψηλού επιπέδου, που απλοποιεί τη διαδικασία υποβολής αιτημάτων HTTP και χειρισμού απαντήσεων. Αναπτύσσεται και συντηρείται από την Square και έχει γίνει η βιβλιοθήκη που χρησιμοποιείται περισσότερο απ’ όλες τις άλλες βιβλιοθήκες τέτοιου τύπου, λόγω της απλότητας, της αποτελεσματικότητας και της στιβαρότητάς της.

Το OkHttp είναι μια βιβλιοθήκη χαμηλού επιπέδου για την εκτέλεση HTTP αιτημάτων, που επίσης αναπτύσσεται και συντηρείται από την Square. Η βιβλιοθήκη αυτή προσφέρει βασικές δυνατότητες, όπως την αποστολή αιτημάτων GET, POST, PUT και DELETE, και την επιστροφή των αποκρίσεων. Είναι ουσιαστικά ένας HTTP client που διαχειρίζεται αιτήματα και αποκρίσεις.

Το Retrofit, ουσιαστικά, χρησιμοποιεί το OkHttp στο παρασκήνιο για να εκτελέσει τα αιτήματα HTTP. Όταν ορίζονται αιτήματα στο Retrofit, το ίδιο αναθέτει στο OkHttp την πραγματική αποστολή και λήψη δεδομένων. Έτσι, το Retrofit επωφελείται από τα χαρακτηριστικά του OkHttp, όπως την υποστήριξη σύνδεσης SSL, την αποθήκευση cache, και τη διαχείριση αποσυνδέσεων.

Το Retrofit προτιμάται κατά την εργασία με REST APIs και χρειάζεται η διαχείριση πολύπλοκων αιτημάτων και αποκρίσεις JSON.

4.1.4 DataStore

Το DataStore είναι μια νέα και βελτιωμένη λύση αποθήκευσης δεδομένων στο Android, που στοχεύει στην αντικατάσταση του SharedPreferences storage. Βασισμένο στα Kotlin Coroutines και Flow, το DataStore παρέχει δύο διαφορετικές υλοποιήσεις: Proto DataStore, που επιτρέπει την αποθήκευση τυποποιημένων αντικειμένων (υποστηριζόμενα από protocol buffers) και Preferences DataStore, που αποθηκεύει ζεύγη κλειδιών-τιμών. Τα δεδομένα αποθηκεύονται ασύγχρονα, με συνέπεια και συναλλακτικά (transactionally), ξεπερνώντας αρκετά από τα μειονεκτήματα του SharedPreferences.

4.1.5 Tensorflow Lite

Το TensorFlow Lite (συχνά αναφέρεται ως TFLite) είναι μια βιβλιοθήκη της Google που έχει σχεδιαστεί κυρίως για ενσωματωμένες συσκευές όπως τα κινητά. Αυτό χρησιμοποιεί έναν προσαρμοσμένο κατανεμητή μνήμης για καθυστέρηση εκτέλεσης και ελάχιστο φορτίο. Το TensorFlow Lite παίρνει τα υπάρχοντα μοντέλα TensorFlow και τα μετατρέπει σε μια βελτιστοποιημένη έκδοση με είδος αρχείου .tflite. Με απλότητα, δημιουργεί εφαρμογές μηχανικής μάθησης για συσκευές iOS και Android.

Το Tensorflow Lite επιτρέπει την εύκολη εκτέλεση μοντέλων μηχανικής μάθησης σε ένα smartphone, επιτρέποντας την εκτέλεση παραδοσιακών εργασιών μηχανικής μάθησης χωρίς την ανάγκη εξωτερικού API ή διακομιστή. Ως αποτέλεσμα, τα μοντέλα θα λειτουργούν σε συσκευές που δεν είναι συνδεδεμένες στο διαδίκτυο.

4.2 .NET Core 8

Το .NET Core, το οποίο συχνά πλέον αναφέρεται, απλά ως .NET, είναι ένα ανοιχτού κώδικα διαπλατφορμικό (cross-platform) framework που έχει δημιουργηθεί από την Microsoft. Το 2014 η Microsoft ανακοίνωσε το .NET Core ως τον αντικαταστάτη του .NET Framework και δημοσίευσε τον πηγαίο κώδικα για την υλοποίηση του .NET Core (CoreCLR), τον πηγαίο κώδικα για ολόκληρη τη στοίβα βιβλιοθηκών για το .NET Core, και ανακοίνωσε την υιοθέτηση ενός συμβατικού μοντέλου ανάπτυξης ανοικτού κώδικα υπό την κηδεμονία του .NET Foundation. Το .NET Core είναι σχεδιασμένο για την ανάπτυξη σύγχρονων εφαρμογών σε διάφορα λειτουργικά συστήματα, συμπεριλαμβανομένων των Windows, macOS και Linux. Σε αντίθεση με το παραδοσιακό .NET Framework, το .NET Core είναι δομημένο με αρθρωτό τρόπο, επιτρέποντας στους προγραμματιστές να συμπεριλάβουν μόνο τις βιβλιοθήκες που χρειάζονται για την ανάπτυξη της εφαρμογής τους, μειώνοντας έτσι το συνολικό μέγεθος των εφαρμογών. Το .NET Core υποστηρίζει μια ενιαία βάση κώδικα που επιτρέπει τη δημιουργία εφαρμογών. Μεταξύ των εφαρμογών που μπορούν να δημιουργηθούν με την χρήση του .NET Core είναι Web εφαρμογές, Desktop, Mobile, Cloud, Video Games, IoT και Machine Learning. Μερικά από τα κύρια συστατικά του και εργαλεία που διαθέτει είναι:

- **ASP.NET Core:** Ένα υψηλών επιδόσεων web framework, για τη δημιουργία web διαφόρων εφαρμογών, με αρχιτεκτονικές όπως, MVC, Web APIs και πραγματικού χρόνου με SignlaR
- **Entity Framework Core (EF Core):** Ένα ORM εργαλείο που επιτρέπει και διευκολύνει την ενσωμάτωση βάσης δεδομένων στην εφαρμογή
- **Dotnet CLI:** Μια γραμμή εντολών που επιτρέπει τη δημιουργία, διαχείριση και εκτέλεση εφαρμογών από το τερματικό

- **Nuget Package Manager:** Παρέχει τη δυνατότητα εγκατάστασης βιβλιοθηκών και πακέτων τρίτων και της διαχείρισης εξαρτήσεων, επεκτείνοντας τις δυνατότητες των εφαρμογών.

Πέραν των παραπάνω το .NET υποστηρίζει τις τελευταίες εκδόσεις των C# και F#, ενσωματώνοντας σύγχρονα χαρακτηριστικά τους όπως async/await, pattern matching και records. Αξιοποιεί πολύ αποδοτικά την τεχνική του Dependency Injection, που είναι πολύ σημαντική για τον διαχωρισμό της λογικής υλοποίησης από τις εξαρτήσεις. Είναι βελτιστοποιημένο για αρχιτεκτονικές cloud και microservices (Azure, AWS) ενώ υποστηρίζει και containerization (Docker). Είναι σχεδιασμένο με έμφαση στην απόδοση. Κάθε έκδοση περιλαμβάνει βελτιστοποιήσεις που μειώνουν τον χρόνο εκκίνησης, βελτιώνουν την αποδοτικότητα μνήμης και μειώνουν το κόστος εκτέλεσης, ενώ το ASP.NET Core κατατάσσεται στα ταχύτερα web frameworks στη διαχείριση αιτημάτων HTTP, καθιστώντας το εξαιρετικά αποδοτικό για την υλοποίηση APIs.

Από πλευράς ασφάλειας το ASP.NET Core υποστηρίζει πρότυπα πρωτόκολλα ελέγχου ταυτότητας. Διαθέτει ενσωματωμένα χαρακτηριστικά που βοηθούν στην προστασία των εφαρμογών από cross-site scripting (XSS) και cross-site request forgery (CSRF). Το ASP.NET Core παρέχει μια ενσωματωμένη βάση δεδομένων χρηστών με υποστήριξη για έλεγχο ταυτότητας πολλαπλών παραγόντων και εξωτερικό έλεγχο ταυτότητας με Google, X και άλλα.

Για την ανάπτυξη εφαρμογών .NET Core, τα πιο δημοφιλή περιβάλλοντα ανάπτυξης είναι το Visual Studio IDE, που αποτελεί ένα ολοκληρωμένο περιβάλλον ανάπτυξης, διαθέτοντας μια μεγάλη σειρά ενσωματωμένων εργαλείων και υπηρεσιών που διευκολύνουν τον προγραμματιστή και το Visual Studio Code, το οποίο είναι ένας πολύ ελαφρύς code editor υλοποίησης της Microsoft και παρέχει εργαλεία υποστήριξης γλώσσας και εργαλεία – βοηθήματα με τη μορφή επεκτάσεων (extensions). Υπάρχουν και άλλα IDEs για την ανάπτυξη εφαρμογών σε .NET όπως, το Rider της JetBrains.

4.3 Microsoft SQL Server 2022

Ο Microsoft SQL Server είναι ένα σύστημα διαχείρισης σχεσιακών βάσεων δεδομένων (RDBMS). Ως διακομιστής βάσεων δεδομένων μπορεί να αποθηκεύει και να ανακτά δεδομένα όταν δέχεται αιτήματα από άλλες εφαρμογές στον ίδιο υπολογιστή ή σε κάποιον απομακρυσμένο υπολογιστή αξιοποιώντας το μοντέλο πελάτη-εξυπηρετητή. Η Microsoft παρέχει APIs για την πρόσβαση στον SQL Server μέσω του διαδικτύου ως web υπηρεσία. Ο SQL Server της Microsoft είναι διαθέσιμος για τα λειτουργικά συστήματα Windows και Linux. Μπορεί να χρησιμοποιηθεί και σε άλλα λειτουργικά συστήματα μέσω Linux containers με χρήση Docker.

Ο SQL Server βασίζεται στο σχεσιακό μοντέλο και επιβάλλει την αναφορική ακεραιότητα μεταξύ των αντικειμένων για τη διατήρηση της συνοχής των δεδομένων. Όπως και με άλλες σχεσιακές βάσεις δεδομένων, για τη διατήρηση της ακεραιότητας των δεδομένων, εφαρμόζονται οι αρχές της ατομικότητας, της συνέπειας, της απομόνωσης συναλλαγών (transactions) και της διάρκειας, συλλογικά γνωστές ως ιδιότητες ACID.

4.4 Docker

Το Docker είναι μια πλατφόρμα λογισμικού ανοιχτού κώδικα που υλοποιεί Εικονικοποίηση (Virtualization) σε επίπεδο Λειτουργικού Συστήματος. Ουσιαστικά το Docker προσφέρει αυτοματοποιημένες διαδικασίες για την ανάπτυξη εφαρμογών σε απομονωμένες Περιοχές Χρήστη (User Spaces) που ονομάζονται Software Containers και απλά αναφέρονται συχνότερα ως Containers. Το λογισμικό χρησιμοποιεί τεχνολογίες του πυρήνα του Linux όπως τα cgroups και οι χώροι ονομάτων πυρήνα (kernel namespaces), για να επιτρέπει σε ανεξάρτητα software containers

να εκτελούνται στο ίδιο λειτουργικό σύστημα. Έτσι αποφεύγεται η χρήση επιπλέον υπολογιστικών πόρων που θα απαιτούσε μια εικονική μηχανή (virtual machine).

Το Docker κέρδισε τη δημοτικότητά του χάρη στις δυνατότητες που προσφέρει στην ανάπτυξη και την εγκατάσταση λογισμικού. Ακολουθούν μερικοί από τους κύριους λόγους για τους οποίους το docker έγινε πολύ δημοφιλές:

- **Φορητότητα**: Το Docker διευκολύνει τους προγραμματιστές στο πακετάρισμα των εφαρμογών τους με όλες τις εξαρτήσεις τους σε ένα ενιαίο ελαφρύ container. Διευκολύνει τη διασφάλιση της σταθερής απόδοσης σε διαφορετικά υπολογιστικά περιβάλλοντα.
- **Αναπαραγωγιμότητα**: Εξασφαλίζει ότι οι ρυθμίσεις λογισμικού παραμένουν συνεπείς σε όλα τα περιβάλλοντα ανάπτυξης, δοκιμής και παραγωγής.
- **Αποδοτικότητα**: Το Docker μέσω της αρχιτεκτονικής του που βασίζεται σε containers βελτιστοποιεί τη χρήση των πόρων. Επιτρέπει στους προγραμματιστές να εκτελούν τις πολλαπλές απομονωμένες εφαρμογές σε ένα μόνο σύστημα.
- **Επεκτασιμότητα**: Τα χαρακτηριστικά επεκτασιμότητας του Docker έδωσαν τη δυνατότητα στους προγραμματιστές να απλοποιήσουν το χειρισμό των εφαρμογών τους κατά τη στιγμή της αύξησης του φόρτου εργασίας. Γι’ αυτό και το Docker είναι αποτελεί ιδανικό εργαλείο για τη δημιουργία microservices.

Ακολουθούν ορισμένα από τα βασικά συστατικά του Docker:

- **Docker Engine**: Πρόκειται για ένα βασικό τμήμα του docker, το οποίο χειρίζεται τη δημιουργία και τη διαχείριση των containers.
- **Docker Image**: Είναι ένα πρότυπο, μόνο για ανάγνωση, που χρησιμοποιείται για τη δημιουργία containers, το οποίο περιέχει τον κώδικα της εφαρμογής και τις εξαρτήσεις της.
- **Docker Hub**: Πρόκειται για ένα αποθετήριο που βασίζεται στο cloud και χρησιμοποιείται για την εύρεση και την κοινή χρήση των εικόνων για δημιουργία containers.
- **Dockerfile**: Πρόκειται για ένα script αρχείο που περιέχει οδηγίες για τη δημιουργία μιας εικόνας.
- **Docker Registry**: Είναι ένα σύστημα αποθήκευσης διανομής για εικόνες, όπου μπορούν να αποθηκευτούν οι εικόνες τόσο δημόσια όσο και σε ιδιωτικα.

4.5 TensorFlow

Το TensorFlow είναι μια πλατφόρμα μηχανική μάθησης, ανοικτού κώδικα, με χρήση γραφημάτων ροής δεδομένων. Οι κόμβοι του γράφου αντιπροσωπεύουν μαθηματικές πράξεις, ενώ οι ακμές του γράφου αντιπροσωπεύουν τους πολυδιάστατους πίνακες δεδομένων (tensors) που ρέουν μεταξύ τους. Αυτή η ευέλικτη αρχιτεκτονική επιτρέπει στους αλγορίθμους μηχανικής μάθησης να περιγράφονται ως γράφοι συνδεδεμένων πράξεων. Μπορούν να εκπαιδευτούν και να εκτελεστούν σε GPUs, CPUs και TPUs σε διάφορες πλατφόρμες χωρίς την ανάγκη ανασύνταξης κώδικα, από φορητές συσκευές έως επιτραπέζιους υπολογιστές και διακομιστές υψηλής τεχνολογίας. Αυτό σημαίνει ότι προγραμματιστές όλων των υποβάθρων μπορούν να χρησιμοποιούν τα ίδια σύνολα εργαλείων για να συνεργάζονται, ενισχύοντας σημαντικά την αποδοτικότητά τους.

Το σύστημα αναπτύχθηκε αρχικά από την ομάδα Google Brain Team για τους σκοπούς της διεξαγωγής έρευνας για τη μηχανική μάθηση και τα βαθιά νευρωνικά δίκτυα (DNN). Το σύστημα αυτό όμως αρκετά γενικό ώστε να μπορεί να εφαρμοστεί και σε μια μεγάλη ποικιλία άλλων τομέων.

Υπάρχουν τρία διακριτά μέρη που καθορίζουν τη ροή εργασίας του TensorFlow. Η προ-επεξεργασία των δεδομένων, η κατασκευή του μοντέλου και η εκπαίδευση του μοντέλου για την πραγματοποίηση προβλέψεων. Αξιοποιώντας την αρχιτεκτονική του TensorFlow, η εκπαίδευση μπορεί να πραγματοποιηθεί σε έναν τυπικό προσωπικό υπολογιστή ή σε ένα κέντρο δεδομένων (data center). Τα εκπαιδευμένα μοντέλα μπορούν στη συνέχεια να εκτελούνται σε μια σειρά από πλατφόρμες, από επιτραπέζιους υπολογιστές, έως κινητά και μέχρι και το cloud.

Το TensorFlow μπορεί να χρησιμοποιηθεί για την ανάπτυξη μοντέλων μηχανικής ή βαθιάς μάθησης για διάφορες εργασίες, συμπεριλαμβανομένης της επεξεργασίας φυσικής γλώσσας (NLP), της αναγνώρισης εικόνας (Image Recognition), της αναγνώρισης γραφής και διαφόρων προσομοιώσεων που βασίζονται σε υπολογισμούς, όπως οι μερικές διαφορικές εξισώσεις.

Τα βασικά πλεονεκτήματα του TensorFlow έγκεινται στην ικανότητά του να εκτελεί λειτουργίες χαμηλού επιπέδου σε πολλές πλατφόρμες επιτάχυνσης, στον αυτόματο υπολογισμό των κλίσεων, στην επεκτασιμότητα σε επίπεδο παραγωγής και στη διαλειτουργική εξαγωγή γραφημάτων.

4.6 Keras

Το Keras είναι ένα υψηλού επιπέδου API που εκτελείται πάνω από το TensorFlow. Το Keras προάγει τις αφαιρέσεις (abstractions) του TensorFlow παρέχοντας ένα απλοποιημένο API που προορίζεται για τη δημιουργία μοντέλων για κοινές περιπτώσεις χρήσης. Βασίζεται στις αρχές της φιλικότητας προς τον χρήστη, της συμβατότητας με την Python και την ικανότητα να χρησιμοποιείται σε διάφορες συσκευές και πλατφόρμες. Η κινητήρια ιδέα πίσω από το API αυτό είναι η δυνατότητα μετάβασης από την ιδέα σε ένα αποτέλεσμα σε όσο το δυνατόν λιγότερο χρόνο.

5. Σχεδιασμός Υλοποίησης

5.1 Εφαρμογή Android

Η ανάπτυξη της εφαρμογής Android έχει πραγματοποιηθεί με τη χρήση της γλώσσας προγραμματισμού Kotlin. Η Kotlin είναι πλέον η προτεινόμενη, από την Google, γλώσσα για ανάπτυξη εφαρμογών στο Android, όπως αναφέρθηκε και νωρίτερα. Επομένως, όταν γίνεται η αρχικοποίηση ενός νέου project στο Android είναι και η προεπιλεγμένη γλώσσα σύνταξης κώδικα. Σαν στόχος έκδοσης του Android SDK τέθηκε το API 34 (Android 14), ενώ σαν ελάχιστο υποστηριζόμενο Android SDK τέθηκε το API 24 (Android 7.0). Αυτό σημαίνει ότι οι εφαρμογή έχει σχεδιαστεί για να υποστηρίζεται σε σύγχρονες εφαρμογές android, μέχρι και αρκετά πιο παλιές, οι οποίες τρέχουν το Android 7.0 (Nougat) και φτάνουν σε χρονολογία πίσω στο 2017.

Οι απαραίτητες βιβλιοθήκες για την λειτουργία της εφαρμογής, οι οποίες εγκαταστάθηκαν και εμφανίζονται στο build.gradle (App) αρχείο, είναι οι εξής:

- **Navigation Component**
  libs.androidx.navigation.fragment.ktx
  libs.androidx.navigation.ui.ktx

- **Lifecycle – Coroutines**
  libs.androidx.lifecycle.extensions
  libs.kotlinx.coroutines.core
  libs.kotlinx.coroutines.android
  libs.androidx.lifecycle.viewmodel.ktx

- **Location**
  libs.play.services.location

- **Dagger – Hilt**
  libs.hilt.android
  kapt libs.hilt.android.compiler
  libs.hilt.android
  kapt libs.androidx.hilt.compiler

- **Retrofit**
  libs.retrofit
  libs.converter.gson
  libs.logging.interceptor

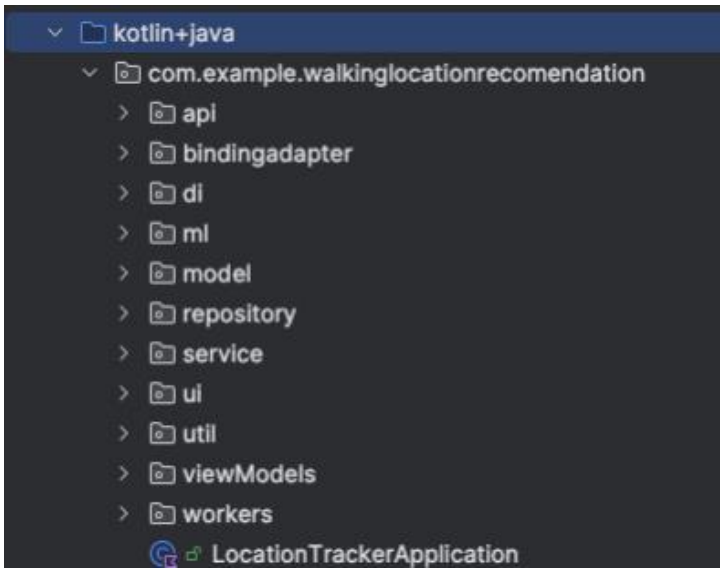
- **OkHttp**
  libs.okhttp
  libs.logging.interceptor.v4100

- **DataStore**
  libs.androidx.datastore.preferences

- **Tensorflow Lite**
  libs.tensorflow.lite
  libs.tensorflow.lite.gpu
  libs.tensorflow.lite.support

Οι περισσότερες από τις βιβλιοθήκες έχουν περιγραφεί στις ενότητες 4.1.1 - 4.1.4. Μια βιβλιοθήκη που δεν έχει περιγράφει είναι το NavigationComponent, μια ισχυρή βιβλιοθήκη που παρέχεται από το Android Jetpack και απλοποιεί την υλοποίηση του navigation μέσα στις εφαρμογές android. Μπορεί να διαχειριστεί σύνθετη πλοήγηση, animations μεταβάσεων από οθόνη σε οθόνη, βαθιά σύνδεση (deep-linking) και ελεγχόμενη σε χρόνο μεταγλώττισης μεταβίβαση παραμέτρων μεταξύ των οθονών μιας εφαρμογής. Άλλη μια βιβλιοθήκη που αξιοποιήθηκε, απαρτίζοντας σημαντικό ρόλο στην υλοποίηση της εφαρμογής και δεν εχει περιγραφεί, είναι το Location SDK από τα play services της Google και παρέχει όλες τις δυνατότητες για ανίχνευση, πρόσβαση και καταγραφή τοποθεσίας.

Κατά τον σχεδιασμό και υλοποίηση της εφαρμογής, τα διαφορετικά αρχεία Kotlin (Classes, Objects) κατανεμήθηκαν σε φακέλους/packages, ανάλογα με την λειτουργία και τον σκοπό τους. Για παράδειγμα τα Activities και τα Fragments έχουν τοποθετηθεί σε ένα πακέτο με όνομα ui μιας και είναι οι κλάσεις που είναι υπεύθυνες για τη δημιουργία και διαχείρηση του γραφικού περιβάλλοντος. Με αυτόν τον τρόπο υπάρχει καλύτερη οργάνωση και καθαρότητα μέσα στη δομή του project. Παρακάτω φαίνεται η δομή του project σε φακέλους.



5.1.1 Δομή φακέλων του Android project

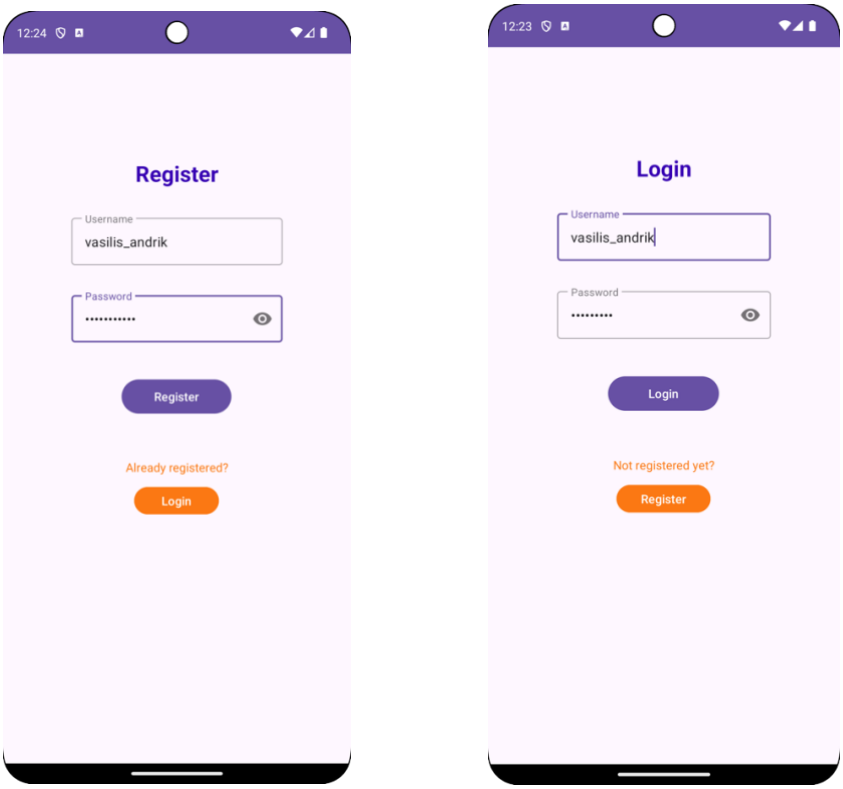
Για την περιγραφή των οθονών της εφαρμογής, καθώς και των μεταβάσεων μεταξύ αυτών, η ανάλυση θα ξεκινήσει από το “ui” package και τα περιεχόμενά του. Στο ui package λοιπόν, υπάρχουν όλες οι κλάσεις που συντελούν στην σύνθεση όλης της γραφικής διεπαφής της εφαρμογής. Συγκεκριμένα στο ui package περιλαμβάνονται η MainActivity, το RegisterFragment, το LoginFragment, το LoadingScreenFragment, το PermissionFragment, το MapsFragment και το ResultsFragment.

Η MainActivity φιλοξενεί δύο διαφορετικούς γράφους πλοήγησης (Navigation Graphs). Η MainActivity αποτελεί βασικό δομικό στοιχείο της εφαρμογής, μιας και διαχειρίζεται τα Fragments που συνθέτουν τα γραφικά στοιχεία της εφαρμογής καθώς και τις μεταβάσεις από τον έναν γράφο πλοήγησης στον άλλον. Αρχικοποιεί ένα NavHostFragment, στο οποίο ενσωματώνονται τα fragments που αποτελούν τις οθόνες της εφαρμογής. Μέσω του NavHostFragment παρέχεται και ο NavController ο οποίος κάνει δυνατή την φόρτωση διαφορετικών γράφων πλοήγησης. Το προκαθορισμένο layout της MainActivity σχηματίζει μια οθόνη φόρτωσης, η οποία αξιοποιείται κατάλληλα τόσο κατά την έναρξη (ή συνέχιση από παύση) της εφαρμογής, όσο και κατά τη μετάβαση από ένα γράφο πλοήγησης σε κάποιον άλλο, μέχρι να φορτωθούν δεδομένα τα οποία ανακτώνται ασύγχρονα, όπως η θερμοκρασία που εμφανίζεται στο MapsFragment.

Ο πρώτος γράφος πλοήγησης (auth_graph) ορίζει τις μεταβάσεις για τα fragments που λαμβάνουν μέρος στις διαδικασίες εγγραφής (RegisterFragment) και εισόδου (LoginFragment) χρηστών στην εφαρμογή. Ως σημείο έναρξης έχει τεθεί το LoginFragment. Το LoginFragment παρέχει μια φόρμα εισόδου ζητώντας από το χρήστη να εισάγει το όνομα χρήστη του και τον κωδικό πρόσβασης. Όταν λοιπόν ένας χρήστης εισάγει αυτά τα στοιχεία μπορεί να πατήσει το κουμπί login για να συνδεθεί στην εφαρμογή. Στο LoginFragment υπάρχει επίσης ένα κουμπί, με την ενεργοποίηση του οποίου, ένας χρήστης μπορεί να μεταβεί στο RegisterFragment.

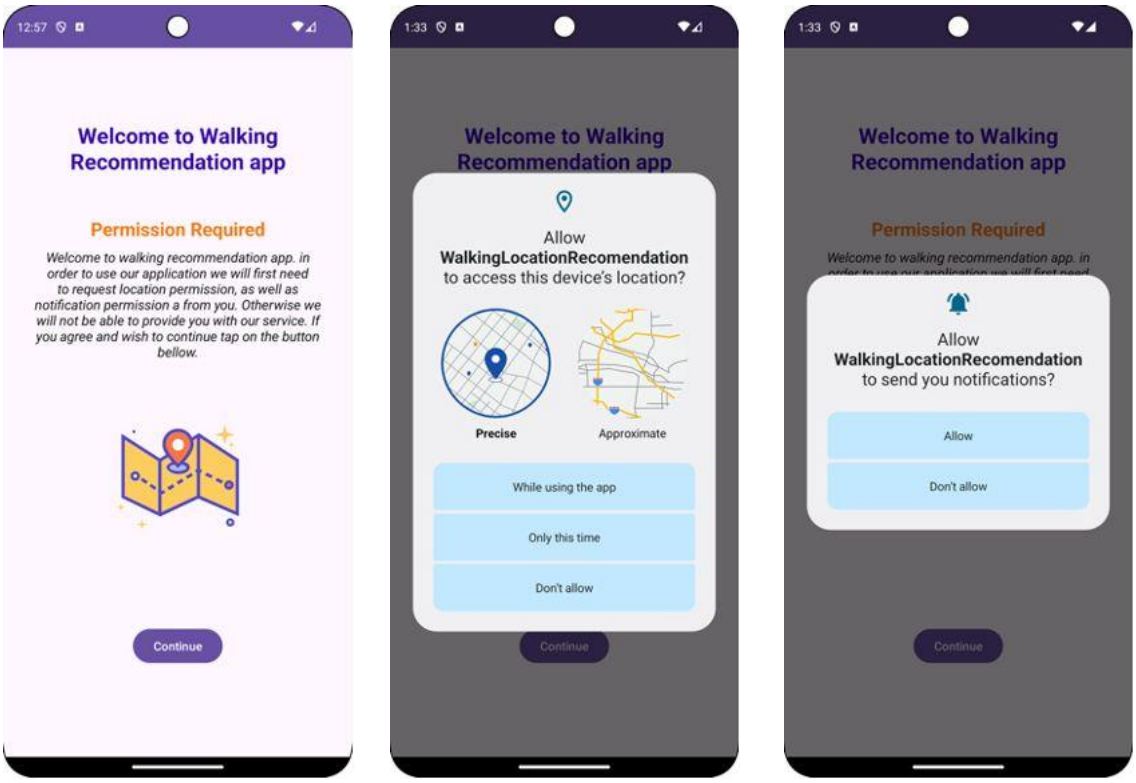
Το RegisterFragment με την σειρά του, παρέχει μια φόρμα εγγραφής χρήστη. Για την εγγραφή ενός χρήστη στην εφαρμογή απαιτούνται από αυτόν να εισαχθούν το όνομα χρήστη του, ο κωδικός πρόσβασης που επιθυμεί και μια επαλήθευση του κωδικού πρόσβασης. Αντίστοιχα με το LoginFragment, όταν ένας χρήστης έχει εισάγει όλα τα απαραίτητα πεδία και πατήσει το κουμπί Register, τότε ολοκληρώνεται η εγγραφή του. Πάλι σε αντιστοιχία με το LoginFragment, στο RegisterFragment υπάρχει ένα κουμπί για την μετάβαση από αυτό στο LoginFragment.

Τα κουμπιά που ανακατευθύνουν το χρήστη από το ένα Fragment στο άλλο χρησιμοποιούν τα actions που έχουν οριστεί στο γράφο πλοήγησης. Η λειτουργία τέτοιων γράφουν που μπορούν να δημιουργηθούν χάρη στο NavigationComponent, προσφέρουν μεγάλη ευκολία στην υλοποίηση μεταβάσεων από ένα fragment σε άλλο. Με τον τρόπο αυτόν λοιπόν, που περιεγράφηκε παραπάνω, ένας χρήστης μπορεί να συνδεθεί πολύ εύκολα στην εφαρμογή εάν είναι εγγεγραμμένος, ενώ αν δεν είναι εγγεγραμμένος μπορεί το ίδιο εύκολα να προβεί στην εγγραφή του.



5.1.2 Οθόνες Εγγραφής και Σύνδεσης του χρήστη στην εφαρμογή, αντίστοιχα

Κατά την σύνδεση ενός χρήστη στην πλατφόρμα, είτε απευθείας από το LoginFragment είτε αφού περάσει πρώτα το στάδιο της εγγραφής από το RegisterFragment, αποθηκεύονται εκ μέρους του στην εφαρμογή δύο tokens, ένα access token και ένα refresh token. Η MainActivity παρατηρεί (observes) τα δύο αυτά tokens. Όταν υπάρχει αποθηκευμένο ένα έγκυρο access token, τότε αντικαθιστά τον πρώτο γράφο πλοήγησης με τον δεύτερο (main_nav_graph) άλλον, ο οποίος παρέχει τη σύνδεση, μέσω μεταβάσεων, μεταξύ των κύριων οθονών της εφαρμογής. Σε αυτόν το γράφο, ως σημείο έναρξης έχει τεθεί το PermissionsFragment. Αυτό το fragment παρουσιάζει πληροφορίες στο χρήστη για τις άδειες (permissions) που πρέπει να παραχωρήσει στην εφαρμογή ώστε να μπορέσει να την χρησιμοποιήσει.



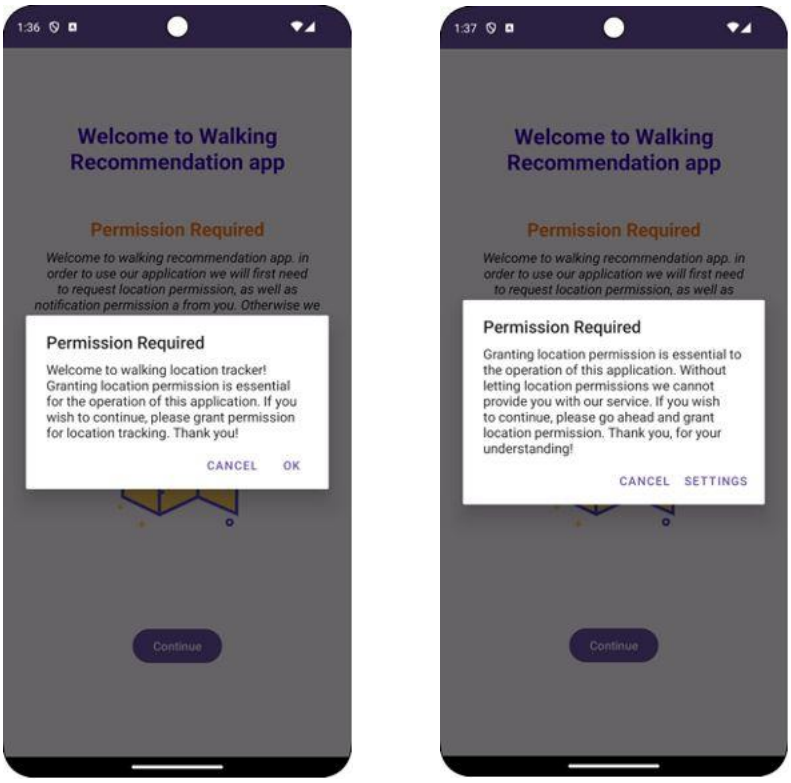
5.1.3 Οθόνη επεξήγησης των αδειών που ζητούνται από το χρήστη και οθόνες διαχείρισης των αδειών για την ανίχνευσης της τοποθεσίας και της εμφάνισης ειδοποιήσεων στο χρήστη αντίστοιχα

Το Android προτείνει να μην γίνεται αποκλεισμός του χρήστη από την εφαρμογή στην περίπτωση που δεν έχει παραχωρήσει ο ίδιος τις απαιτούμενες άδειες, περιορίζοντας απλά το πλήθος δυνατοτήτων που του προσφέρει. Ωστόσο για τις ανάγκες της συγκεκριμένης εφαρμογής, απαιτούνται άδειες που είναι συνδεδεμένες αυστηρά με προσωπικά δεδομένα, καθώς και η αποθήκευση και η επεξεργασία αυτών. Για παράδειγμα η καταγραφή και αποθήκευση της τοποθεσίας του χρήστη αποτελεί δεδομένο που δεν μπορεί να παραληφθεί από το μοντέλο λειτουργίας αυτής της εφαρμογής.

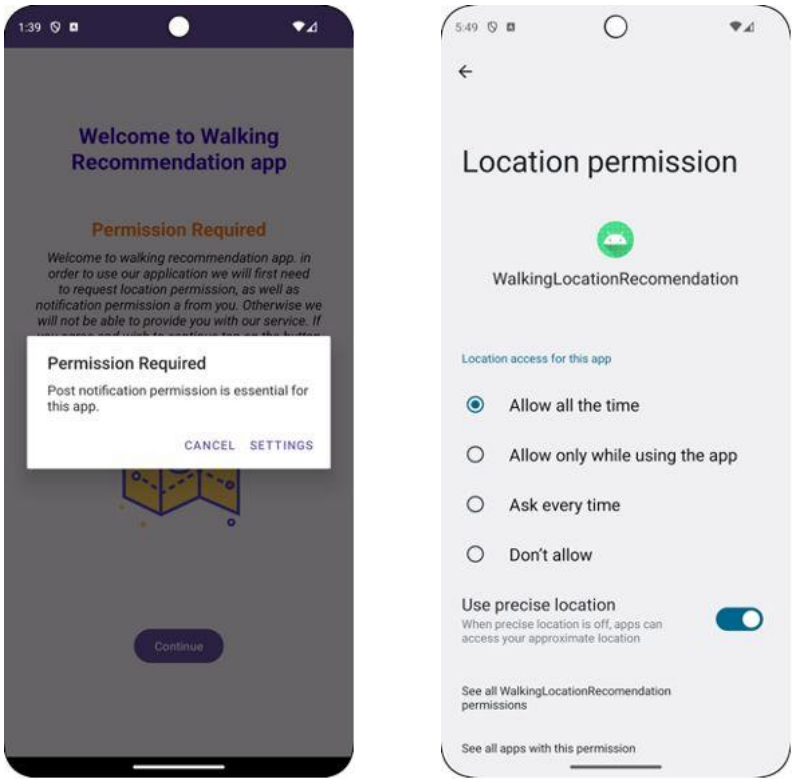
```
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission android:name="android.permission.INTERNET" />

<uses-permission android:name="android.permission.ACCESS_BACKGROUND_LOCATION" />

<uses-permission android:name="android.permission.FOREGROUND_SERVICE" />
<uses-permission android:name="android.permission.FOREGROUND_SERVICE_LOCATION" />
<uses-permission android:name="android.permission.POST_NOTIFICATIONS"/>
```

5.1.4 Οι άδειες που απαιτούνται προς παραχώρηση από το χρήστη και έχουν δηλωθεί στο android_manifest

Επομένως, σε περίπτωση που ο χρήστης δεν δώσει έγκριση για την παραχώρηση των απαραίτητων αδειών, εμφανίζεται ένα dialog που του εξηγεί τον λόγο που πρέπει να παραχωρήσει την κάθε άδεια και τον προτρέπει εκ νέου να παραχωρήσει τις άδειες, ανοίγοντας το dialog που είναι προσχεδιασμένο από το android για την κάθε άδεια. Αν ο χρήστης αρνηθεί πάλι να δώσει έγκριση για τις άδειες που του ζητούνται, τότε εμφανίζεται ένα τελευταίο dialog που τον ενημερώνει ότι δεν μπορεί να προχωρήσει στην εφαρμογή, ενώ αν αλλάξει γνώμη και θέλει να παραχωρήσει τις άδειες, μπορεί να το πράξει από την εφαρμογή ρυθμίσεων της συσκευής του.



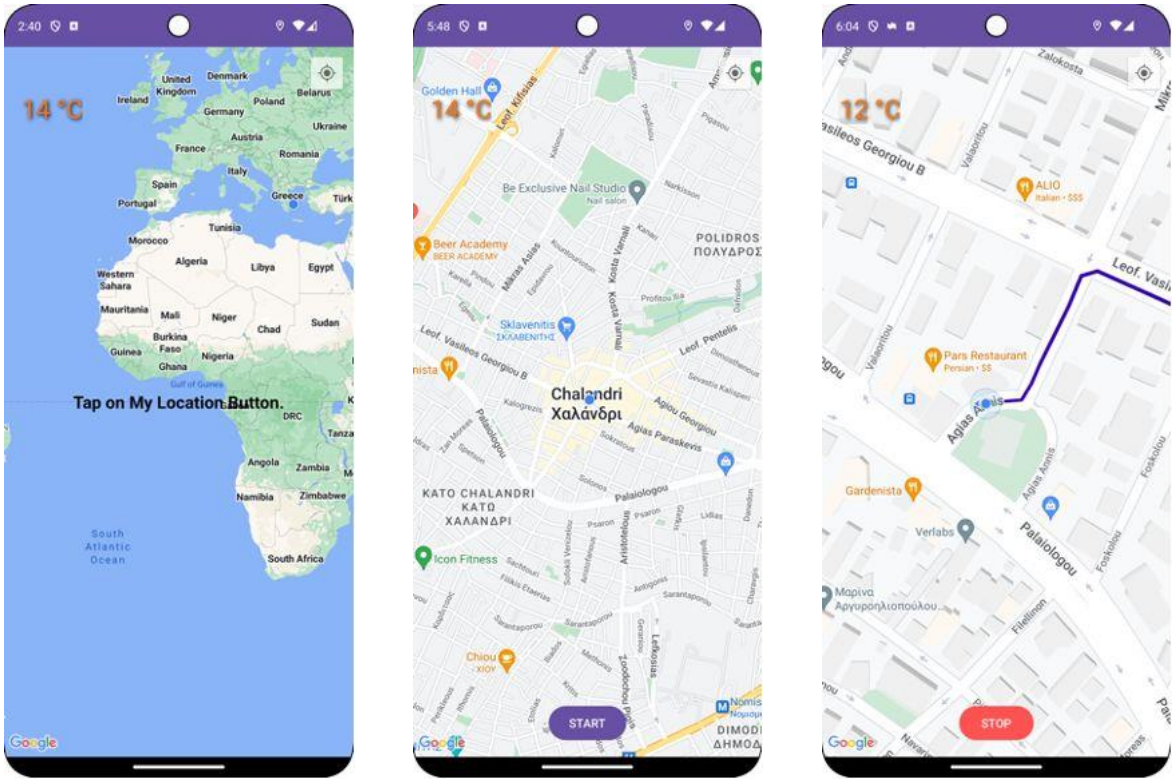
5.1.5 Dialogs που περιγράφουν τους λόγους για τους οποίους πρέπει ο χρήστης να παραχωρήσει τις απαιτούμενες άδειες ώστε να προχωρήσει στην εφαρμογή



5.1.6 Παρότρυνση του χρήστη να μεταβεί στις ρυθμίσεις της συσκευής του ώστε να παραχωρήσει τις απαιτούμενες άδειες στην περίπτωση που έχει αρνηθεί να τις παραχωρήσει

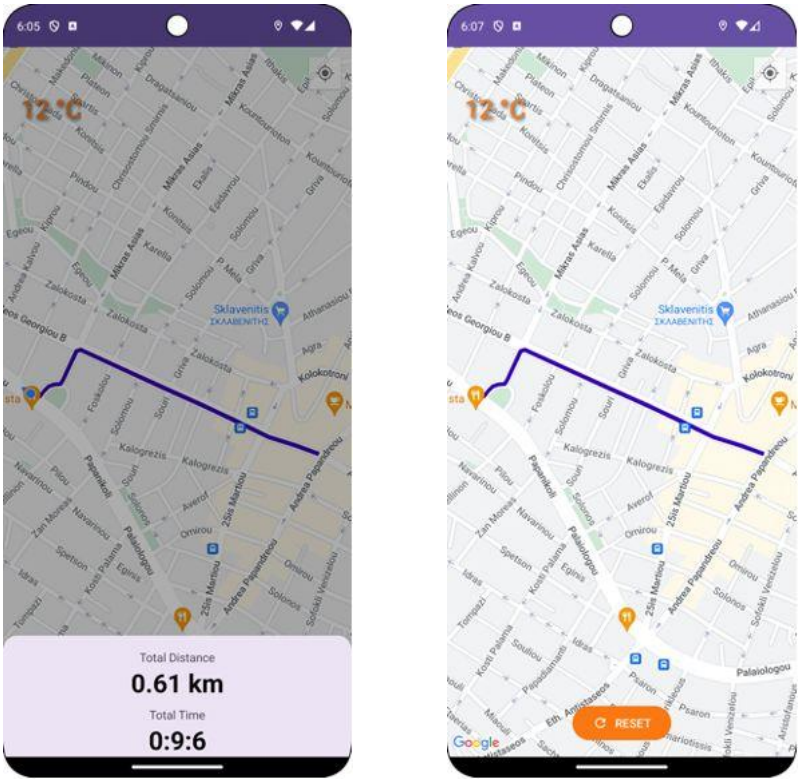
Κατόπιν της αποδοχής και παραχώρησης όλων των απαιτούμενων αδειών, ο χρήστης μεταφέρεται στο επόμενο fragment, το οποίο είναι το MapsFragment. Το MapsFragment αποτελεί την οθόνη στην οποία λαμβάνουν χώρα οι κυριότερες λειτουργίες της εφαρμογής. Σε αυτήν την οθόνη εμφανίζεται ένας χάρτης περιορισμένης διάδρασης με το χρήστη, μιας και είναι κλειδωμένες οι λειτουργίες μεγέθυνσης και περιστροφής της κάμερας.

Όταν ο χρήστης ξεκινάει σε αυτήν την οθόνη, υπάρχει ένα κείμενο που τον προτρέπει να πατήσει το κουμπί της τοποθεσίας του. Στο πάνω μέρος αυτής της οθόνης, επιπλέον, εμφανίζεται και η τρέχουσα θερμοκρασία στην τοποθεσία του. Όταν ο χρήστης πατήσει το κουμπί τοποθεσίας, τότε ο χάρτης κάνει zoom στην τοποθεσία του χρήστη κι εμφανίζεται στο κάτω μέρος της οθόνης ένα κουμπί με την ένδειξη “start”. Πατώντας αυτό το κουμπί ο χρήστης ξεκινάει την διαδρομή του και παράλληλα και την καταγραφή δεδομένων κατά τη διάρκεια αυτής. Για όσο χρόνο βρίσκεται σε διαδρομή ο χρήστης, την θέση του κουμπιού με την ένδειξη start λαμβάνει ένα κουμπί κόκκινου χρώματος με την ένδειξη “stop”. Επιπρόσθετα, στην οθόνη εμφανίζεται η γραμμή που απεικονίζει τη διαδρομή του χρήστη και η κάμερα εστιάζει στην εκάστοτε τοποθεσία του. Την στιγμή που ο χρήστης θελήσει να τερματίσει τη διαδρομή του, αρκεί να πατήσει το κουμπί “stop”.



5.1.7 Οθόνη χάρτη πριν ο χρήστης πατήσει τον εντοπισμό της τοποθεσίας του, πριν ξεκινήσει μια διαδρομή και αφού έχει ξεκινήσει μια διαδρομή αντίστοιχα

Κατά τον τερματισμό της διαδρομής ένα ακόμη fragment εμφανίζεται, πάνω στο MapsFragment, το ResultsFragment το οποίο εμφανίζει μερικά στατιστικά δεδομένα της διαδρομής του χρήστη. Μέσα σε αυτά τα στατιστικά είναι ο χρόνος διάρκειας της διαδρομής του χρήστη και η συνολική απόσταση που κάλυψε. Όταν ο χρήστης βρίσκεται σε αυτήν την οθόνη, αν πατήσει σε κάποιο σημείο εκτός της, τότε αυτή «κρύβεται» και ο έλεγχος επιστρέφει στο MapsFragment. Σε αυτό το στάδιο στο MapsFragment στην θέση πλέον του κουμπιού “stop” βρίσκεται ένα νέο κουμπί με ένδειξη “reset”. Στο πάτημα αυτού του κουμπιού, διαγράφεται από την οθόνη η γραμμή απεικόνισης της διαδρομής του χρήστη και το κουμπί “reset” αλλάζει σε “start”. ΄Ετσι ο χρήστης είναι έτοιμος να ξεκινήσει μια νέα διαδρομή.



5.1.8 Οθόνη αποτελεσμάτων αφού ο χρήστης τερματίσει την διαδρομή του και οθόνη χάρτη αφού ο χρήστης κλείσει την οθόνη αποτελεσμάτων αντίστοιχα

Σε αυτό το σημείο έχουν περιγράφει όλες κλάσεις που δημιουργούν το γραφικό περιβάλλον της εφαρμογής, καθώς και η λειτουργία της καθεμίας από αυτές. Στη συνέχεια, θα περιγράψουμε τις σημαντικότερες από τις λειτουργικότητες που έχουν υλοποιηθεί, καθώς και το πώς αυτές ενεργούν συνδυαστικά και συντελούν την λειτουργία της εφαρμογής.

Όπως αναφέρθηκε νωρίτερα, για την χρήση της εφαρμογής από χρήστες είναι απαραίτητη η έλευση πρώτα από το στάδιο της αυθεντικοποίησης. Για την λειτουργία της αυθεντικοποίησης, είναι απαραίτητη η χρήση και αποθήκευση ενός ζεύγους tokens, ένα access token και ένα refresh token. Αυτά τα δύο tokens δημιουργούνται για τον χρήστη όταν αυτός κάνει login στην εφαρμογή και αποθηκεύονται στο DataStore. Το DataStore, όπως αναλύθηκε νωρίτερα, αντικαθιστά τα shared preferences storage του Android. Στην εφαρμογή χρησιμοποιείται το Preferences DataStore, το οποίο αποθηκεύει δεδομένα σε ζεύγη μορφής κλειδί-τιμή. Όπως εξηγείται στο documentation του Android, για το DataStore πρέπει να δημιουργείται και να επαναχρησιμοποιείται αυστηρά και μόνο ένα instance του, ειδάλλως μπορεί να προκύψουν διαφόρων ειδών προβλήματα. Για τον λόγο αυτό γίνεται αρχικοποίηση του DataStore στην MainActivity κι έπειτα αυτό χρησιμοποιείται στις κλάσεις που το χρειάζονται, μέσω της τεχνικής του dependency injection στον constructor, με τη χρήση του Hilt.

```
/**
 * DataStore Instance. Should be the only one in the project.
 * So wee need to treat it as a singleton
 */
private val Context.dataStore: DataStore<Preferences> by preferencesDataStore(name = "settings")

@AndroidEntryPoint  VasilisAndrikopoulos *
class MainActivity : AppCompatActivity() {
```

5.1.9 Δήλωση του DataStore instance στον κώδικα της MainActivity

Επιστρέφοντας, στην αποθήκευση των tokens στην εφαρμογή, έχει γίνει υλοποίηση μιας κλάσης AuthDataStoreRepository η οποία υλοποιεί το interface DataStoreRepository. Το DataStorerepository παρέχει τα access token και refresh token σε μορφή Flow. Το Flow είναι ένα reactive stream API των Coroutines, σχεδιασμένο για την εκτέλεση ασύγχρονων εργασιών που παράγουν δεδομένα με ροή. Είναι ένας τρόπος χειρισμού ροών δεδομένων που μπορούν να μεταφέρουν πολλές τιμές με την πάροδο του χρόνου. Επιπλέον το DataStoreRepository interface ορίζει και δύο μεθόδους, μία για αποθήκευση των tokens και μια την διαγραφή τους.

```
/**
 * DataStoreRepository
 * - Interface for the DataStoreRepository
 * - Provides the access token and refresh token as flows
 * - Provides functions to save and clear tokens
 */
interface DataStoreRepository {  VasilisAndrikopoulos
    val accessToken: Flow<String?>
    val refreshToken: Flow<String?>
    suspend fun saveTokens(accessToken: String, refreshToken: String)
    suspend fun clearTokens()  VasilisAndrikopoulos
}
```

5.1.10 Το DataStoreRepository interface

Στη συνέχεια, έχουμε μια ακόμα κλάση, την AuthRepository, η οποία με τη σειρά της, μέσω της τεχνικής του dependency injection με τη χρήση του Hilt, παίρνει σαν όρισμα στον constructor το DataStoreRepository που περιγράφηκε προηγουμένως και το αξιοποιεί για την υλοποίηση διάφορων λειτουργιών. Μέσα σε αυτές τις λειτουργίες, πέραν των saveTokens() και clearTokens() για τις οποίες καλεί τις αντίστοιχες του DataStoreRepository, υλοποιεί και την ανανέωση του ζεύγους από tokens χρησιμοποιώντας τον authService client, που τον διαθέτει σαν property και τον αρχικοποιεί. Η υλοποίηση του autheService θα αναλυθεί παρακάτω. Η ανανέωση του ζεύγους tokens γίνεται με την μέθοδο refreshTokens(), η οποία ελέγχει αν υπάρχουν ενεργά tokens και σε περίπτωση που δεν διατίθενται τέτοια, κάνει μια κλήση στο Web API που έχει υλοποιηθεί μέσω του AuthService, για να λάβει πίσω ένα καινούριο ζεύγος tokens. Η εν λόγω κλάση χρησιμοποιείται στο AuthViewModel το οποίο είναι το επόμενο μέρος κώδικα που θα αναλυθεί, καθώς και στον AuthInterceptor του οποίου η λογική και η λειτουργία θα αναλυθούν μεταγενέστερα.

Τα ViewModel(s) αποτελούν μέρος των Android Architecture Components και βοηθούν στη διαχείριση δεδομένων που σχετίζονται με τη διεπαφή χρήστη (UI), με τρόπο που λαμβάνει υπ'όψην τον κύκλο ζωής της εφαρμογής. Στα view models έχουμε την κλάση AuthViewModel, η οποία έχει δημιουργηθεί για την αποθήκευση και την ανάκτηση των δύο tokens που έχουν αναφερθεί παραπάνω (access & refresh tokens), ώστε αυτά πλέον να παρέχονται στα Activities σαν live data. Αυτή η κλάση χρησιμοποιείται στην MainActivity, για την παρακολούθηση των tokens ώστε αυτή να μπορεί να ελέγξει την διαχείριση του ui που θα εμφανίσει στο χρήστη. Επιπλέον η AuthViewModel χρησιμοποιείται και στο LoginFragment για την αποθήκευση των tokens στο DataStore, όταν ο χρήστης ολοκληρώσει ένα επιτυχές αίτημα εισόδου και λάβει πίσω τα tokens.

Το επόμενο ζήτημα που θα αναλυθεί είναι η υλοποίηση των κλάσεων που συντελούν στην λήψη και αποθήκευση δεδομένων, τόσο από το Web API του συστήματος, όσο και από το WeatherApi του OpenWeatherMaps για την λήψη δεδομένων που σχετίζονται με τον καιρό.

Η πρώτη υλοποίηση που θα εξετάσουμε είναι αυτή του RetrofitAuthInstance. Πρόκειται για μια κλάση που δημιουργεί έναν Retrofit http client για να επικοινωνεί και να μπορεί να στείλει αιτήματα στο API του συστήματος, για τις λειτουργίες αυθεντικοποίησης του χρήστη. Δημιουργεί ένα Retrofit object με τον Builder που διαθέτει το Retrofit, δίνοντας σαν ορίσματα στον builder, το url του API και έναν GsonConverter, για την μετατροπή της Json απάντησης από το Web API σε Gson. Το Gson (Google's JSON) είναι μια βιβλιοθήκη της Java που δημιουργήθηκε από την Google και έχει σχεδιαστεί ειδικά για την μετατροπή JSON σε Java objects και το αντίστροφο.

```
/**
 * Retrofit instance for the authentication API
 */
private val retrofitAuth by lazy {
    Retrofit.Builder()
        .baseUrl(TRACKING_API_BASE_URL)
        .addConverterFactory(GsonConverterFactory.create())
        .build()
}
```

5.1.11 Δημιουργία του Retrofit object για το authentication

Με τη χρήση του retrofit object που δημιουργήθηκε, στη συνέχεια δημιουργεί ένα instance του authService (client) τύπου IAuthService, τροφοδοτώντας το με μια υλοποίηση του IAuthService interface. Επιπλέον αυτή η κλάση παρέχει και μια μέθοδο για την επιστροφή του authService instance, ώστε να αποτρέπεται η δημιουργία δεύτερου instance του client στην εφαρμογή.

```
/**
 * AuthService instance
 */
private val authService: IAuthService by lazy {
    retrofitAuth.create(IAuthService::class.java)
}

fun getInstance(): IAuthService {  VasilisAndrikopoulos
    return authService
}
```

5.1.12 Δημιουργία του authService client

Το IAuthService είναι ένα interface που ορίζει τα endpoints με τα οποία μπορεί να αλληλεπιδράσει το authService. Γενικά το retrofit, στην δημιουργία ενός client απαιτεί την εισαγωγή υλοποίησης ενός interface το οποίο θα ορίζει τα endpoints του api, με το οποίο αυτός θα μπορεί να αλληλεπιδρά. Τα endpoints ορίζονται σαν συναρτήσεις που πρέπει να μπορούν να λειτουργούν ασύγχρονα (γι' αυτό κι εδώ είναι δηλωμένες ως suspend fun), με ορίσματα τις παραμέτρους ή το σώμα (body) που περιμένει να λάβει το api, καθώς και το http action με μορφή Annotation, συνοδευόμενο από το path του endpoint. Οι παράμετροι query και τα body δεδομένα σημαίνονται και αυτά με τα αντίστοιχα annotations. Για την απάντηση από το api σε κάθε function πρέπει να παρέχεται και ο τύπος επιστροφής. Για τον τύπο επιστροφής Response πρέπει να οριστεί και ο τύπος δεδομένων που αναμένεται να έχει η απάντηση από το api. Οι διαφορετικοί τύποι δεδομένων που δέχεται η εφαρμογή από το api σαν responses συνολικά, καθώς και αυτοί που χρησιμοποιούνται σαν request bodies, έχουν μοντελοποιηθεί σε data classes. Οι data classes σε συνδυασμό με το annotation @SerializedName που παρέχεται από τη βιβλιοθήκη Gson, βοηθούν στην μετατροπή της εκάστοτε απάντησης από το api σε Kotlin classes και το αντίστροφο, από Kotlin class σε Json body.

```
interface IAuthService {  VasilisAndrikopoulos
    @POST("Auth/register")  VasilisAndrikopoulos
    suspend fun registerUser(@Body user: User): Response<UserResponse>

    @POST("Auth/login")  VasilisAndrikopoulos
    suspend fun login(@Body user: User): Response<Tokens>

    @POST("Auth/refresh-token")  VasilisAndrikopoulos
    suspend fun refreshToken(@Body refreshTokenRequest: Tokens): Response<Tokens>
}
```

5.1.13 Η υλοποίηση του IAuthService interface με τις μεθόδους που απαιτούνται για την επικοινωνία με τα αντίστοιχα endpoints του Web API

Επομένως, το authService παρέχει τη διασύνδεση με το Web API του συστήματος για τις λειτουργίες της εγγραφής, της σύνδεσης του χρήστη, καθώς και της ανανέωσης του ζεύγους από tokens που πρέπει να διαθέτει. Αυτός ο client χρησιμοποιείται όπως αναφέρθηκε παραπάνω από την κλάση AuthRepository και από τα LoginFragment και RegisterFragment για την αλληλεπίδραση με το Web API και την διεξαγωγή ενεργειών λήψη δεδομένων που σχετίζονται με την αυθεντικοποίηση.

Τώρα, θα αναλυθεί το RetrofitTrackingModule ένα (Kotlin) Object που έχει υλοποιηθεί για την αλληλεπίδραση της android εφαρμογής με το Web API του συστήματος, ώστε αυτή να μπορεί να στέλνει τα δεδομένα των χρηστών που έχει καταγράψει, σε μορφή logs και να αποθηκευτούν στη βάση δεδομένων. Ο λόγος που αυτό το Module έχει σχεδιαστεί σαν object και όχι σαν κλάση, είναι ότι για την αποστολή δεδομένων στο Web API χρειάζεται η χρήση ενός AuthInterceptor.

Στις εφαρμογές που ακολουθούν το μοντέλο Πελάτη-Διακομιστή, interceptors παρεμβάλλονται μεταξύ requests και responses επιτελώντας κάποιο έργο. Στην συγκεκριμένη εφαρμογή, για να μπορέσει ένας χρήστης να στείλει δεδομένα τα οποία θα αποθηκευτούν στη βάση, πρέπει να είναι συνδεδεμένος. Ο χρήστης της εφαρμογής θεωρείται συνδεδεμένος, όταν διαθέτει ένα έγκυρο access token. Όταν λοιπόν, γίνει ένα αίτημα αποθήκευσης δεδομένων στο Web API, αλλά το access token του χρήστη δεν είναι έγκυρο (ο πιο πιθανός λόγος είναι να έχει λήξει), τότε το Web API στέλνει απάντηση με status code 401, δηλαδή ότι ο χρήστης δεν είναι συνδεδεμένος στο σύστημα και συνεπώς δεν έχει πρόσβαση να πράξει την αποθήκευση δεδομένων.

Ο AuthInterceptor που έχει υλοποιηθεί, επεκτείνει (extends) την κλάση Interceptor της βιβλιοθήκης okhttp και κατά συνέπεια επεκτείνει και την μέθοδο intercept αυτής της κλάσης. Η μέθοδος intercept του AuthInterceptor παίρνει σαν παράμετρο μια αλυσίδα (chain) από requests/responses και έχει σαν τύπο επιστροφής Response. Η μέθοδος αυτή έχει επεκταθεί ώστε να κάνει τα εξής. Πρώτα ανακτά το access token του χρήστη που είναι αποθηκευμένο στο DataStore και το προσθέτει στο “Authorization” header του request ως Bearer token, στη συνέχεια προσθέτει το request στην αλυσίδα και το εκτελεί κι έπειτα περιμένει την απάντηση (response). Αν η απάντηση έχει status 401 (Unauthenticated), τότε στέλνει νέο request στο Web API για να δημιουργήσει ένα καινούριο ζεύγος κλειδιών (tokens). Αν πετύχει η δημιουργία κλειδιών τότε τα αποθηκεύει στο DataStore και στη συνέχεια χρησιμοποιεί το νέο access token για να εκτελέσει πάλι το πρώτο request. Με την επιτυχία της εκ νέου εκτέλεσης του request για αποθήκευση δεδομένων, επιστρέφει τελικά την απάντηση. Αν από την άλλη αποτύχει η δημιουργία νέου ζεύγους κλειδιών, διαγράφει τα ήδη υπάρχοντα και αναγκάζει το χρήστη να κάνει login. Αν το αρχικό request αποτύχει για άλλο λόγο, επιστρέφει την απάντηση και αφήνει τον έλεγχο του χειρισμού της στην μέθοδο που έκανε την κλήση εξαρχής. Με αυτόν τον τρόπο καταφέρνουμε να ανανεώνουμε την είσοδο του χρήστη στην πλατφόρμα χωρίς να τον αναγκάζουμε να κάνει login, αν το token του δεν είναι πια έγκυρο κατά τη διάρκεια μιας διαδρομής, με σκοπό να μην χαθούν τα δεδομένα που κατέγραψε η συσκευή του.

Επομένως, η απάντηση στο γιατί αυτός ο client, για την αποθήκευση δεδομένων, υλοποιήθηκε σαν module σε Kotlin Object δίνεται από την περιγραφή του AuthInterceptor που δόθηκε νωρίτερα. Ο AuthInterceptor χρειάζεται να μπορεί να ανακτά, να αποθηκεύει και να διαγράφει τα tokens του χρήστη και αυτό γίνεται μέσω του AuthRepository, που εισέρχεται στον constructor του interceptor με τη χρήση του Hilt και dependency injection και αυτό με τη σειρά του λαμβάνει στον constructor του το DataStoreRepository με τον ίδιο τρόπο, σε μια διαδικασία που περιγράφηκε αρκετά παραπάνω. Αυτό σημαίνει ότι δεν μπορεί να γίνεται δημιουργία στιγμιοτύπων του AuthInterceptor αυθαίρετα και αυτός θα πρέπει να παρέχεται με κάποιον τρόπο σαν singleton. Τα @Module, @Provide και @Singleton Annotations που προσφέρει η βιβλιοθήκη dagger δίνουν τη δυνατότητα, για την υλοποίηση αυτή. Έτσι, το RetrofitTrackingModule παρέχει μέσω συναρτήσεων τον AuthInterceptor, τον OkHttpClient ο οποίος χρησιμοποιείται για την τροποποίηση των προκαθορισμένου okHttpClient που χρησιμοποιεί το Retrofit, ώστε να μπορεί να ενσωματώσει και τον AuthInterceptor, το retrofit instance για τη δημιουργία του tracking client και σαφώς και τον ίδιο τον client που μπορεί να χρησιμοποιηθεί για κλήσεις στο Web API, προς αποθήκευση δεδομένων. Οι συναρτήσεις αυτές όμως δεν μπορούν να χρησιμοποιηθούν απευθείας στον κώδικα της εφαρμογής για την εξυπηρέτηση του τελικού σκοπού που είναι η αποθήκευση δεδομένων στο Web API.

Μια κλάση που έχει δημιουργηθεί με όνομα TrackingRepository, παρέχει την δυνατότητα αποθήκευσης μέσω μιας ασύγχρονης συνάρτησης addLog() με παράμετρο τα δεδομένα που πρόκειται να αποθηκευτούν. Το Hilt, στο παρασκήνιο συνδυάζει όλες αυτές τις συναρτήσεις του RetrofitTrackingModule και τελικά παράγει ένα instance του ITrackingService το οποίο εισάγεται με dependency injection στον constructor της κλάσης TrackingRepository. Με τη σειρά του το TrackingRepository γίνεται inject στο σημείο της εφαρμογής όπου πραγματοποιείται η αποστολή δεδομένων και το σημείο αυτό η ανάλυση θα γίνει μετέπειτα.

```kotlin
@Module  VasilisAndrikopoulos
@InstallIn(SingletonComponent::class)
object RetrofitTrackingModule {


    @Provides  VasilisAndrikopoulos
    @Singleton
    fun provideAuthInterceptor(authRepository: AuthRepository): AuthInterceptor {
        return AuthInterceptor(authRepository)
    }

    @Provides  VasilisAndrikopoulos
    @Singleton
    fun provideOkHttpClient(authInterceptor: AuthInterceptor): OkHttpClient {
        return OkHttpClient.Builder()
            .addInterceptor(authInterceptor)
            .addInterceptor(HttpLoggingInterceptor().apply {
                level = HttpLoggingInterceptor.Level.BODY // Logs request and response bodies
            })
            .connectTimeout( timeout: 30, TimeUnit.SECONDS)
            .readTimeout( timeout: 30, TimeUnit.SECONDS)
            .writeTimeout( timeout: 30, TimeUnit.SECONDS)
            .build()
    }

    @Provides  VasilisAndrikopoulos
    @Singleton
    fun provideRetrofitTrackingInstance(okHttpClient: OkHttpClient): Retrofit {
        return Retrofit.Builder()
            .baseUrl(TRACKING_API_BASE_URL)
            .addConverterFactory(GsonConverterFactory.create())
            .client(okHttpClient)
            .build()
    }

    @Provides  VasilisAndrikopoulos
    @Singleton
    fun provideTrackingService(retrofit: Retrofit): ITrackingService {
        return retrofit.create(ITrackingService::class.java)
    }
}
```

5.1.14 Η υλοποίηση του RetrofitTrackingModule που δίνει τη δυνατότητα δημιουργίας του trackingService μέσω DI με τη βοήθεια του Dagger – Hilt

Πέραν των client(s) που δημιουργήθηκαν για την αλληλεπίδραση με το Web API του συστήματος, υλοποιήθηκε κι άλλος ένας client ο οποίος χρησιμεύει στην λήψη δεδομένων καιρού βασιζόμενο στην τοποθεσία, από το OpenWeatherMaps API. Πολλές σύγχρονες συσκευές android διαθέτουν θερμόμετρο, υγρασιόμετρο και άλλους αισθητήρες που μπορούν να καταγράφουν δεδομένα από το περιβάλλον. Ωστόσο, αυτή τη στιγμή είναι μικρό το εύρος των συσκευών που διαθέτουν τέτοιους αισθητήρες.Συνεπώς, για τις ανάγκες της συγκεκριμένης εφαρμογής, η υλοποίηση δεν βασίστηκε σε τέτοιου τύπου αισθητήρες, αλλά σε ένα πλήρες και ακριβές API που παρέχει τέτοιες πληροφορίες όπως η θερμοκρασία, οι καιρικές συνθήκες με μορφή ετικέτας, η ταχύτητα του ανέμου, το ποσοστό υγρασίας και άλλα. Η υλοποίηση του client για τη λήψη δεδομένων από το OpenWeatherMaps API δε διαφέρει ουσιαστικά από αυτήν του authService με τη διαφορά ότι εδώ γίνεται κλήση σε ένα API τρίτου. Το εν λόγω API για δεδομένη τοποθεσία, επιστρέφει ένα αρχείο json με μια μεγάλη πληθώρα διαφορετικών δεδομένων, εκ των οποίων διατηρήθηκαν κάποια συγκεκριμένα για τις ανάγκες τις εφαρμογής.

Και σε αυτήν την περίπτωση χρειάστηκε η υλοποίηση ενός Kotlin Module με διαφορετικά data classes για την μετατροπή των δεδομένων από JSON σε Kotlin classes. Σε αυτήν την περίπτωση, επίσης έχει δημιουργηθεί ένα WeatherRepository Kotlin Object για την ευκολότερη αξιοποίηση του weatherService μέσω ενός και μόνο function με όνομα getWeatherData() που δέχεται σαν παραμέτρους την τοποθεσία του χρήστη, ένα appId και δυο ακόμα που αποτελούν ρυθμίσεις προτιμήσεων για τις μονάδες μέτρησης και τη γλώσσα.

Με τα παραπάνω ολοκληρώθηκε η ανάλυση και των τμημάτων υλοποίησης επικοινωνίας με τα APIs που χρησιμοποιούνται. Επόμενο μέρος της εφαρμογής που θα αναλυθεί ως προς τη δομή και τη λειτουργία του είναι το TrackerService. Το TrackerService είναι από τα σημαντικότερα τμήματα της android εφαρμογής, αν όχι το πιο σημαντικό.

Στο Android, τα Services είναι στοιχεία που εκτελούν μεγάλες σε διάρκεια λειτουργίες στο παρασκήνιο χωρίς άμεση αλληλεπίδραση με τη διεπαφή χρήστη. Είναι ιδανικά για εργασίες που δεν απαιτούν αλληλεπίδραση με τον χρήστη, αλλά πρέπει να συνεχίσουν να εκτελούνται ακόμη και όταν η εφαρμογή βρίσκεται στο παρασκήνιο. Το συγκεκριμένο πρόκειται για ένα Foreground Service και ως τέτοιο, πραγματοποιεί λειτουργίες για τις οποίες ο χρήστης παραμένει ενεργά ενημερωμένος συνεχώς. Τέτοιου τύπου Services απαιτούν την ύπαρξη μιας μόνιμης ειδοποίησης (notification) η οποία δεν κλείνει αν δεν τερματιστεί το Service. Συνεπώς αν ο χρήστης προσπαθήσει να κλείσει την εφαρμογή, η λειτουργία αυτής καθώς και η ύπαρξη της ειδοποίησης δεν τερματίζουν. Αν ο χρήστης θελήσει να τερματίσει την εφαρμογή πρέπει πρώτα να τερματίσει το service.

Η επιλογή ενός τέτοιου Service έχει πραγματοποιηθεί με τη λογική ότι, για όσο χρόνο ο χρήστης κάνει μια διαδρομή και το Service καταγράφει δεδομένα γι’ αυτόν, πρέπει η εφαρμογή να μένει επίσης ενεργή. Με αυτόν τον τρόπο διασφαλίζεται το ότι τα δεδομένα που έχουν καταγραφεί θα αποσταλούν όταν τερματίσει τη διαδρομή του ο χρήστης και δεν θα χαθούν, αλλά ούτε και θα γίνεται καταγραφή δεδομένων επ’ αόριστων, χωρίς την συνειδητή του επιλογή. Επιπλέον, η ύπαρξη της ενεργής ειδοποίησης, προσφέρει στο χρήστη τη δυνατότητα προβολής της απόστασης που έχει καλύψει από την ώρα που ξεκίνησε η διαδρομή του.

Συνεχίζοντας με την τεχνική ανάλυση του service, πριν αναλυθούν συγκεκριμένες λεπτομέρειες υλοποίησης, αυτό ξεκινά όπως περιεγράφηκε νωρίτερα, όταν ο χρήστης βρίσκεται στο MapsFragment και αφού έχει πατήσει το κουμπί εντοπισμού της τοποθεσίας του, πατήσει και το κουμπί “start”. Την πρώτη φορά που ο χρήστης χρησιμοποιεί την εφαρμογή και θα πατήσει το κουμπί “start”, δε θα ξεκινήσει το service, αντ’ αυτού η εφαρμογή θα τον μεταφέρει στην εφαρμογή ρυθμίσεων της συσκευής του, ώστε αυτός να παραχωρήσει μια ακόμη άδεια που σχετίζεται με την τοποθεσία. Πρόκειται για την άδεια ACCESS_BACKGROUND_LOCATION και ο χρήστης από τις ρυθμίσεις θα πρέπει να επιλέξει την ένδειξη για χρήση της τοποθεσίας συνεχώς. Αυτή η άδεια απαιτείται για να μπορεί να γίνει δυνατή η καταγραφή της τοποθεσίας του από το service ακόμα και αν η οθόνη της συσκευής του είναι κλειστή. Μόλις ο χρήστης δώσει αυτήν την άδεια, μπορεί να ξεκινήσει τη διαδρομή του πατώντας το κουμπί “start” εκ νέου. Μόλις λοιπόν,ο χρήστης πατήσει το συγκεκριμένο κουμπί τότε ξεκινάει και το foreground service.

Στον κώδικα του service υπάρχει ένα companion object στο οποίο είναι δηλωμένες κάποιες παρατηρήσιμες και τροποποιήσιμες μεταβλητές (MutableLiveData). Αυτές οι μεταβλητές είναι οι εξής: η μεταβλητή started (boolean) που υποδηλώνει το αν έχει ξεκινήσει το service, η μεταβλητή startTime που δηλώνει την στιγμή έναρξης του service σε milliseconds (Long), η μεταβλητή stopTime που δηλώνει την στιγμή τερματισμού του service σε milliseconds (Long) και οι locationList, temperatureList, weatherLabelList, brightnessList που είναι λίστες οι οποίες διατηρούν δεδομένα καταγραφής καθ’όλη τη διάρκεια λειτουργίας του service, για τις τοποθεσίες του χρήστη, τις θερμοκρασίες στις τοποθεσίες αυτές, τις καιρικές συνθήκες στις τοποθεσίες αυτές και τις τιμές φωτεινότητας της συσκευής του αντίστοιχα.

Κατά τη δημιουργία του service, στην onCreate(), αρχικοποιεί αυτές τις MutableLiveData μεταβλητές και έναν fusedLocationProviderClient για την καταγραφή τοποθεσιών. Στην μέθοδο onStartCommand() του service περνάει μια ενέργεια από το κουμπί “start” με όνομα ACTION_SERVICE_START και τότε εκτελούνται μια μέθοδος startForegroundService() η οποία δημιουργεί την ενεργή ειδοποίηση και καλεί την startForeground() μέθοδο που κληρονομείται από την κλάση Service, και μια μέθοδο startLocationUpdates(). Η startLocationUpdates() καλεί την μέθοδο requestLocationUpdates() του fusedLocationProviderClient δίνοντας της σαν όρισμα ένα locationRequest και ένα locationCallback. Η requestLocationUpdates() ζητάει ουσιαστικά την τοποθεσία του χρήστη ανά τακτά χρονικά διαστήματα. Το locationRequest δημιουργείται επίσης στην startLocationUpdates() με τον Builder της κλάσης LocationRequest, που προέρχεται από τη location βιβλιοθήκη της Google. Στο locationRequest έχουν τεθεί ουσιαστικά κάποιες προτιμήσεις, για την ακρίβεια και τη συχνότητα των requests που γίνονται, για την εύρεση της τοποθεσίας του χρήστη την εκάστοτε στιγμή. Το locationCallback είναι ένα object μιας ανώνυμης κλάσης που επεκτείνει την LocationCallback και κάνει override την μέθοδο onLocationResult(). Ουσιαστικά στην onLocationResult() γίνονται κάποιες διαδικασίες όταν υπάρχει ένα καινούριο location σαν αποτέλεσμα της requestLocationResults().

Συγκεκριμένα, κάθε φορά που καλείται η onLocationResult() γίνεται η καταγραφή των δεδομένων του χρήστη σε μεταβλητές/λίστες. Γίνεται επίσης μια κλήση στο OpenWeatherMaps API για την λήψη δεδομένων καιρού για την τρέχουσα τοποθεσία του χρήστη. Έτσι αποθηκεύονται η τοποθεσία του χρήστη, η θερμοκρασία στην τοποθεσία του, οι καιρικές συνθήκες ως ετικέτα, για παράδειγμα “Clear sky” και η φωτεινότητα της οθόνης από τη συσκευή του χρήστη στην τρέχουσα στιγμή. Επίσης γίνεται ανανέωση του κειμένου της ειδοποίησης, το οποίο δείχνει την απόσταση που έχει διανύσει ο χρήστης και με κάθε καινούρια τοποθεσία που προστίθεται αυτή αυξάνεται.

Την στιγμή που ο χρήστης πατήσει το κουμπί “stop” από το MapsFragment και ενώ πραγματοποιεί κάποια διαδρομή, τότε μια εντολή μεταβιβάζεται στο service, μέσα στην onStartCommand() μέθοδό του. Η εντολή αυτή πρόκειται για την ACTION_SERVICE_STOP, η οποία υποδεικνύει στο service να τερματίσει. Έτσι καλείται η μέθοδος stopForegroundService() η οποία έχει υλοποιηθεί με τρόπο που εκτελεί κάποιες ενέργειες πριν σταματήσει το service. Με χρήση coroutines υπολογίζει την μέση τιμή για κάθε ένα από τους τύπους δεδομένων που καταγράφονται, με τον τρόπο που θα περιγραφεί στην επόμενη παράγραφο, στέλνει το αποτέλεσμα σαν ένα log στο API του συστήματος προς αποθήκευση, αξιοποιώντας την αντίστοιχη μέθοδο του trackingRepository κι έπειτα καλεί μια ακόμα μέθοδο με όνομα stopService(). Η stopService() απενεργοποιεί την ενεργή ειδοποίηση και κατόπιν καλεί τη stopForeGround() και την stopSelf() που προέρχονται από την κλάση Service, για να τερματίσει τελικά το service.

Ο υπολογισμός των μέσων τιμών που αναφέρθηκε στην προηγούμενη παράγραφο πραγματοποιείται από κάποιες βοηθητικές συναρτήσεις που έχουν υλοποιηθεί με τέτοιον τρόπο ώστε:

α) η τοποθεσία υπολογίζεται ως το «κέντρο» της διαδρομής που έχει διανύσει ο χρήστης, υπολογίζοντας τον μέσο όρο για καθένα από τα latitude και longitude για το πλήθος των τοποθεσιών που βρίσκονται στην locationList

β) η θερμοκρασία υπολογίζεται ως ο μέσος όρος των θερμοκρασιών που έχουν καταγραφεί κατά τη διαδρομή του, σε κάθε διαφορετική τοποθεσία

γ) για την «ετικέτα» του καιρού υπολογίζεται η ενδιάμεση τιμή μιας και αποτελεί κατηγορικό δεδομένο

δ) η φωτεινότητα της οθόνης υπολογίζεται ως ο μέσος όρος καταγραφών φωτεινότητας

ε) ο συνολικός αριθμός βημάτων υπολογιζόμενος μέσω της απόστασης που διένυσε, θεωρώντας ότι κατά μέσο όρο το βήμα ενός ανθρώπου έχει απόσταση 0.75 μέτρα

Πέραν αυτών των δεδομένων, στο log που αποστέλλεται στο Web API προστίθεται και μια χρονοσφραγίδα (timestamp) από την οποία προκύπτουν αρκετές πληροφορίες, όπως η ώρα της ημέρας, η ημέρα της εβδομάδας και ο μήνας.

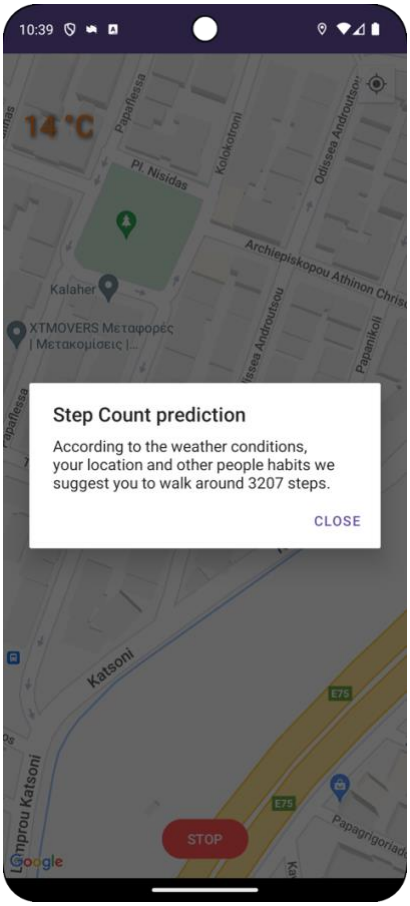
Σε αυτό το σημείο, έχοντας αναλύσει όλα τα τμήματα της εφαρμογής Android σε επίπεδο λειτουργίας και τα κυριότερα τμήματα της εφαρμογής σε τεχνικό επίπεδο, πρωτού η ανάλυση προχωρήσει στο Web API του συστήματος, θα αναφερθούν ονομαστικά κάποια ακόμα μέρη κώδικα που είχαν υποστηρικτικό ρόλο στην υλοποίηση και στην λειτουργία της εφαρμογής. Τα περισσότερα από αυτά αποτελούν objects που παρέχουν βοηθητικές μεθόδους και συναρτήσεις με στατικό τρόπο. Τελικώς θα γίνει και η αναφορά της διαδικασίας ενσωμάτωσης του TensorFlow Lite μοντέλου στην εφαρμογή.

Περνώντας στην αναφορά αυτή, έχει υλοποιηθεί ένα object Constants, το οποίο περιέχει διάφορες σταθερές, όπως permission request codes, στατικό κείμενο επεξήγησης για καθεμιά από τις άδειες που ζητάει η εφαρμογή, urls και άλλες τιμές όπως τα action commands για το service. Άλλο ένα object που έχει υλοποιηθεί είναι αυτό με όνομα Dialogs το οποίο παρέχει την κατασκευή dialogs τα οποία εμφανίζονται στον χρήστη στο PermissionsFragment, ώστε να τον ενημερώσουν για τις άδειες που ζητάει η εφαρμογή και να τον προτρέψουν σε κάποια ενέργεια με την παροχή κουμπιών, όπως “close”, “settings” και άλλα. Αντίστοιχης υλοποίησης object είναι και το Permissions, στο οποίο έχουν υλοποιηθεί βοηθητικές μέθοδοι για την εμφάνιση και την διαχείριση των αδειών. Η διαχείριση των αδειών γίνεται με τη χρήση της registerActivityForResult(), στη οποία ορίζεται και την διαδικασία που ακολουθεί ανάλογα με το αν επιλέξει να παραχωρήσει κάποιες άδειες ή όχι. Η registerActivityForResult() αποτελεί για το android μια μοντέρνα αντικατάσταση για τις μεθόδους startActivityForResult() και onActivityResult(). Συνεχίζοντας, μερικά ακόμα objects που έχουν δημιουργηθεί είναι, το MapUtil στο οποίο υπάρχουν κάποιες βοηθητικές μέθοδοι που σχετίζονται με το MapsFragment, το BrightnessHelper που παρέχει ένα function που επιστρέφει την ακέραια (int) τιμή φωτεινότητας της οθόνης και το ExtensionFunctions που παρέχει επεκτάσεις των UI Classes View και Button.

Κλείνοντας το τμήμα αναφοράς της υλοποίησης της εφαρμογής android, θα γίνει και η αφορά ενσωμάτωσης του Tensorflow Lite μοντέλου στην εφαρμογή. Σε μια διαδικασία που θα περιγραφεί παρακάτω, μέσω της χρήσης εργαλείων που διαθέτει η βιβλιοθήκη του TensorFlow, γίνεται η μετατροπή του μοντέλου TensorFlow σε TensorFlow Lite και εξάγεται σε ένα αρχείο με κατάληξη .tflite. Αυτό το αρχείο εισάγεται στο Android Project στο φάκελο assets. Η φόρτωση του μοντέλου γίνεται στο MapsFragment, με τη μέθοδο loadModelFile(). Η συνάρτηση αυτή επιστρέφει ένα αντικείμενο τύπου MappedByteBuffer. Το MappedByteBuffer χρησιμοποιείται για την είσοδο/έξοδο αρχείων μέσω μνήμης (memory-mapped file I/O). Με αυτόν τον τρόπο, το μοντέλο φορτώνεται εξοικονομώντας μνήμη και επιτρέποντας ταχύτερη πρόσβαση στη διάρκεια της διαδικασίας της αναγνώρισης (inference). Αυτό το αντικείμενο που επιστρέφει η loadModelFile() εισάγεται σαν παράμετρος στον constructor του Interpreter που αρχικοποιεί την μεταβλητή model. Η μεταβλητή model αποτελεί property του MapsFragment και ουσιαστικά μέσω αυτής παρέχεται το μοντέλο, για εισαγωγή δεδομένων και παραγωγή πρόβλεψης. Αφού αρχικοποιηθεί η μεταβλητή model, καλείται η εντολή model.allocateTensors(), που απαιτείται μετά τη δημιουργία του Interpreter, καθώς κατανέμει τη μνήμη για τους εισόδους και εξόδους του μοντέλου (tensors).

Για την δημιουργία της εισόδου του μοντέλου έχει δημιουργηθεί ένα Kotlin Object με όνομα InputTransformer, το οποίο παρέχει διάφορες μεθόδους για την μετατροπή των δεδομένων εισόδου που διατίθενται στην εφαρμογή, σε είσοδο για το μοντέλο TensorFlow Lite. Έτσι για το σύνολο δεδομένων εισόδου γίνεται η μετατροπή τους και η χαρτογράφηση σε ένα FloatArray, που είναι και ο απαιτούμενος τύπος εισόδου για το μοντέλο. Έτσι κατά τη χρήση της εφαρμογής κι όταν ένας χρήστης

ξεκινάει μια καινούρια διαδρομή, πραγματοποιούνται η συλλογή των δεδομένων που χρειάζονται για την είσοδο του μοντέλου και η παραγωγή της πρόβλεψης των βημάτων που θα μπορούσε να πραγματοποιήσει ο χρήστης. Η πρόβλεψη αυτή εμφανίζεται στο χρήστη με τη μορφή ενός Dialog.



5.1.15 Εμφάνιση της πρόβλεψης σε αριθμό βημάτων στο χρήστη με τη μορφή Dialog

5.2 Web API

Το Web API του όλου συστήματος έχει υλοποιηθεί με τη χρήση του .NET Core Framework 8 σε γλώσσα CSharp (C#). Ο ρόλος του Web API είναι να παρέχει μια υποδομή για την αποθήκευση δεδομένων καταγραφής από την εφαρμογή android, καθώς και να παρέχει έναν τρόπο αποθήκευσης αυτών των δεδομένων προσωποποιημένα και ελεγχόμενα μέσω αυθεντικοποίησης. Για την αποθήκευση των δεδομένων, το Web API έχει συνδεθεί με μια βάση δεδομένων σε Microsoft SQL Server. Στο backend σύστημα έχει ενταχθεί η υποστήριξη από Docker containers, έτσι το deployment του backend γίνεται με χρήση docker compose, το οποίο δημιουργεί και τρέχει ένα container για το Web API και άλλο ένα για τον MSSQL Server σε κοινό δίκτυο.

Για την υλοποίηση του Web API σε .NET Core χρησιμοποιήθηκε το περιβάλλον ανάπτυξης Visual Studio της Microsoft, το οποίο παρέχει μια ευρεία γκάμα από εργαλεία υλοποίησης αλλά και επιλογών από εφαρμογές και βιβλιοθήκες. Τόσο για την λειτουργία όσο και για τον σχεδιασμό και την υλοποίηση του Web API, πραγματοποιήθηκε η εγκατάσταση κάποιων NuGet πακέτων.

Αυτά τα NuGet πακέτα είναι τα εξής:

- Microsoft.AspNetCore.Authentication.JwtBearer
- Microsoft.EntityFrameworkCore
- Microsoft.EntityFrameworkCore.SqlServer
- Microsoft.EntityFramowrkCore.Tools
- Bcrypt.Net-Next
- Swashbuckle.AspNetCore
- Swashbuckle.AspNetCore.Filters

Το Swashbuckle.AspNetCore έρχεται ήδη εγκατεστημένο κατά τη δημιουργία ενός Web API στο Visual Studio και παρέχει την αυτόματη δημιουργία api documentation βασισμένο στο OpenAPI (Swagger). Κατά τη λειτουργία της εφαρμογής σε περιβάλλον υλοποίησης καθίσταται δυνατή η προβολή του SwaggerUI για την διεξαγωγή δοκιμών στα endpoints που το API υλοποιεί. Το πακέτο Swashbuckle.AspNetCore.Filters αντικαθιστά το παλαιότερο Swashbuckle.AspNetCore.Examples και επεκτείνει τις λετουργίες του Swashbuckle.AspNetCore παρέχοντας κάποιες ακόμα επιλογές στο documentation και στο SwaggerUI, όπως το να εμφανίζονται API requests/responses ως παραδείγματα και η παροχή ασφάλειας/κλειδώματος σε endpoints που χαρακτηρίζονται από το [Authorize] attribute στους controllers τους.

Το Microsoft.AspNetCore.Authentication.JwtBearer παρέχει ένα middleware component για τον έλεγχο ταυτότητας μέσω JSON Web Token (JWT). Το middleware αυτό λειτουργεί για την επικύρωση JWT tokens που εκδίδονται από το API, εξασφαλίζοντας ασφαλή πρόσβαση στους πόρους του. Το Bcrypt.Net-Next παρέχει μεθόδους για την κρυπτογράφηση δεδομένων. Συγκεκριμένες μέθοδοι από αυτό το πακέτο είναι ιδιαίτερα χρήσιμες για την κρυπτογράφηση των κωδικών πρόσβασης των χρηστών πριν αυτοί να αποθηκευτούν στη βάση. Παρέχει επίσης τη δυνατότητα ελέγχου για το αν ένας κωδικός πρόσβασης ταιριάζει στον θεωρητικά αντίστοιχο κρυπτογραφημένο του στη βάση δεδομένων. Πρακτικά αν ένας χρήστης που θέλει να συνδεθεί στην εφαρμογή εισάγει λάθος τον κωδικό του, ο έλεγχος αυτός θα δείξει ότι ο κωδικός δεν ταιριάζει με τον αντίστοιχο κρυπτογραφημένο στη βάση δεδομένων.

Το Microsoft.EntityFrameworkCore είναι ένα πακέτο που παρέχει το Entity Framework Core. Το Entity Framework Core είναι μια τεχνολογία πρόσβασης δεδομένων που μπορεί να λειτουργεί ως object-relational mapper (ORM) εργαλείο. Αυτό το εργαλείο επιτρέπει την εργασία με μια βάση δεδομένων χρησιμοποιώντας αντικείμενα .NET και εξαλείφει το μεγαλύτερο μέρος του κώδικα πρόσβασης σε δεδομένα που θα χρειαζόταν να γραφτεί. Με το EF Core, η πρόσβαση στα δεδομένα πραγματοποιείται με τη χρήση ενός μοντέλου. Ένα μοντέλο αποτελείται από κλάσεις οντοτήτων και ένα αντικείμενο context που αναπαριστά μια σύνοδο με τη βάση δεδομένων. Το αντικείμενο context επιτρέπει την αναζήτηση και την αποθήκευση δεδομένων. Το Entity Framework υποστηρίζει τις ακόλουθες διαφορετικές προσεγγίσεις ανάπτυξης μοντέλων:

- Δημιουργία μοντέλων από μια υπάρχουσα βάση δεδομένων.
- Χειροκίνητη συγγραφή ενός μοντέλου για να ταιριάζει με τη βάση δεδομένων.
- Μόλις δημιουργηθεί ένα μοντέλο, γίνεται αξιοποίηση των Migrations του Entity Framework για τη δημιουργία μιας βάσης δεδομένων από το μοντέλο. Τα migrations επιτρέπουν την εξέλιξη της βάσης δεδομένων καθώς το μοντέλο αλλάζει.

Τα εργαλεία που επιτρέπουν την δημιουργία του σχήματος της βάσης μέσω reverse engineering του μοντέλου και scaffolding του context με τη μορφή Migrations, παρέχονται από το πακέτο Microsoft.EntityFramowrkCore.Tools. Το Entity Framework Core υποστηρίζει πολλές μηχανές βάσεων δεδομένων, όπως Microsoft SQL, MySQL/MariaDB και PostgresSQL.

Στην συγκεκριμένη υλοποίηση γίνεται η χρήση του Microsoft SQL Server, επομένως και για την υποστήριξή του,έγινε η εγκατάσταση του πακέτου Microsoft.EntityFrameworkCore.SqlServer. Αυτό το πακέτο ουσιαστικά περιέχει τον provider για τον Microsoft SqlServer και τον AzureSql.

Συνεχίζοντας με την ανάλυση της υλοποίησης αυτό το Web API διαθέτει τρία μοντέλα που αντιστοιχούν σε ισάριθμους πίνακες στη βάση δεδομένων. Το πρώτο μοντέλο είναι το μοντέλο User. Το μοντέλο User διαθέτει σαν properties τα Id, UserName και PasswordHash. Το UserName είναι μοναδικό για κάθε χρήστη. Το PasswordHash πρόκειται για την κρυπτογραφημένη μορφή του κωδικού πρόσβασης που εισάγει ο χρήστης κατά την εγγραφή του. Το μοντέλο User διαθέτει και συσχετίσεις ένα-προς-πολλά για τα άλλα δύο μοντέλα που περιγράφονται στη συνέχεια.

Το επόμενο μοντέλο είναι το RefreshToken, το οποίο διαθέτει Id, Token (string), Created (datetime) και Expires (datetime). Το Token είναι το πραγματικό αλφαριθμητικό που αντιστοιχεί στο refresh token, το Created είναι η στιγμή της δημιουργίας και το Expired είναι η στιγμή που λήγει αυτό το token. Το RefreshToken έχει συσχέτιση με το User ένα-προς-ένα, μιας και ένα token ανήκει αποκλειστικά και μόνο σε ένα χρήστη. Ο λόγος που γίνεται χρήση μοντέλου για το RefreshToken και συνεπώς και αποθήκευση του στη βάση δεδομένων, είναι ότι υπάρχει η ανάγκη ταυτοποίησης του χρήστη που το χρησιμοποιεί, καθώς και το αν έχει λήξει, διότι η δημιουργία και το είδος αυτού του token διαφέρει από την υλοποίηση του access token, όπως θα περιγραφεί στη συνέχεια.

Το τελευταίο μοντέλο που έχει σχεδιαστεί είναι το TrackingLog. Αυτό το μοντέλο έχει σαν properties όλα αυτά τα δεδομένα που αποστέλλονται από το android. Απεικονίζει δηλαδή ένα TrackingLog όπως αυτό προκύπτει όταν τερματίζει η διαδρομή του χρήστη στο Foreground Service. Πέραν αυτών των properties διαθέτει και μια συσχέτιση με το μοντέλο User ως ένα-προς-ένα. Δηλαδή ένα TrackingLog ανηκεί σε έναν και μόνο συγκεκριμένο χρήστη.

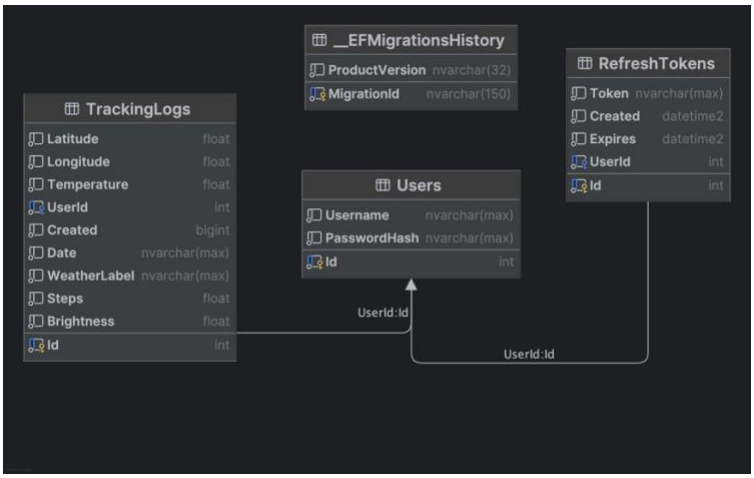
Κατόπιν της δημιουργίας των μοντέλων ακολούθησε η δημιουργία του db context που απαιτεί το Entity Framework Core για την λειτουργία των Migrations όπου θα σχεδιάσει το σχήμα της βάσης καθώς και τις λειτουργείες αποθήκευσης και αναζήτησης δεδομένων από τη βάση με τη χρήση των μοντέλων που περιεγράφηκαν παραπάνω.

Η ApiDbContext κλάση που δημιουργήθηκε επεκτείνει την κλάση DbContext που προέρχεται από το EntityFrameworkCore πακέτο. Στην ApiDbContext δηλώνονται ως DbSet(s) οι οντότητες που αντιστοιχούν στους πίνακες της βάσης και ουσιαστικά αποτελούν συλλογές από το κάθε υλοποιημένο μοντέλο αντίστοιχα. Επομένως, αυτά τα properties επιτρέπουν την εκτέλεση ανάκτησης/αναζήτησης και την αποθήκευση δεδομένων για τις αντίστοιχες οντότητες/μοντέλα.

```
public class ApiDbContext: DbContext
{
    VasilisAndrikopoulos
    public ApiDbContext(DbContextOptions<ApiDbContext> options) : base(options) { }

    3 usages
    public DbSet<User> Users { get; set; }
    public DbSet<RefreshToken> RefreshTokens { get; set; }
    1 usage
    public DbSet<TrackingLog> TrackingLogs { get; set; }
}
```

5.2.1 Η υλοποίηση της κλάσης ApiDbContext

Το σχήμα της βάσης δεδομένων όπως προκύπτει όταν γίνει η εκτέλεση των migrations φαίνεται στην παρακάτω εικόνα. Αποτελείται από τρεις πίνακες που αντιστοιχούν σε κάθε ένα από τα μοντέλα, όπως αναλύθηκε παραπάνω και περιλαμβάνει κι έναν ακόμα πίνακα που διατηρεί το ιστορικό των migrations που έχουν εφαρμοστεί στη βάση δεδομένων.



5.2.2 Το σχήμα της βάσης δεδομένων

Η αυθεντικοποίηση χρηστών όπως έχει αναφερθεί πολλές φορές, είναι σημαντικό κομμάτι της υλοποίησης αυτής της εργασίας και ιδιαίτερα του Web API. Η ταυτοποίηση χρηστών και η διαχείριση των token (access & refresh) επιτυγχάνονται με την χρήση της AuthService κλάσης που αποτελεί υλοποίηση του IAuthService interface.

```
4 usages  1 inheritor  VasilisAndrikopoulos
public interface IAuthService
{
    2 usages  1 implementation  VasilisAndrikopoulos
    string CreateToken(User user);
    2 usages  1 implementation  VasilisAndrikopoulos
    RefreshToken GenerateRefreshToken();
    1 usage  1 implementation  VasilisAndrikopoulos
    ClaimsPrincipal? GetPrincipalFromExpiredToken(string token);
}
```

5.2.3 Το IAuthService το οποίο ορίζει τις μεθόδους για την διαχείριση των tokens

Το IAuthService ορίζει προς υλοποίηση τις τρεις μεθόδους που παρουσιάζονται στην παραπάνω εικόνα και η λειτουργία τους κατά σειρά είναι:

- **CreateToken**: δημιουργεί και επιστρέφει το accessToken για το χρήστη. Στην συγκεκριμένη υλοποίηση της AuthService, δημιουργείται μια λίστα από claims που περιέχει το Username και το Id του χρήστη. Στη συνέχεια δημιουργείται ένα SymmetricSecurityKey με τη χρήση ενός secret το οποίο είναι δηλωμένο στο AppSettings configuration αρχείο. Τέλος, πραγματοποιείται η δημιουργία ενός JWT το οποίο χρησιμοποιεί τα claims ως payload του JWT, το κλειδί που δημιουργήθηκε για

την υπογραφή του jwt και η τρέχουσα ημερομηνία προσθέτοντας μια μέρα, ως η ημερομηνία λήξης του token.

- **GenerateRefreshToken**: δημιουργεί και επιστρέφει το refresh token. Στη συγκεκριμένη υλοποίηση της AuthService φτιάχνει ένα καινούριο object του μοντέλου RefreshToken, όπου το Token είναι ένα base64 string που δημιουργείται και το Expires είναι 7 μέρες από την ημερομηνία δημιουργίας.

- **GetPrincipalFromExpiredToken**: επιστρέφει τα claims (ClaimsPrincipal) ή null για ένα access token (jwt) που έχει λήξει.

Έπειτα το IAuthService interface χρησιμοποιείται στον AuthController. Ο AuthController είναι ο ένας από τους δύο controllers που έχουν υλοποιηθεί. Το IAuthService αποτελεί παράμετρο στον constructor του AuthController. Η υλοποίηση AuthService του IAuthService, επιλέγεται προς χρήση στο παρασκήνιο και για την επίτευξη αυτού του σκοπού η χρήση της συγκεκριμένης υλοποίησης γίνεται χρησιμοποιώντας dependency injection. Αυτό καθίσταται δυνατό εφόσον στο Program.cs αρχείο έχει οριστεί ότι για το IAuthService, θα πρέπει να παρέχεται η AuthService ως κλάση που το υλοποιεί.

Ο AuthController περιέχει τρεις συναρτήσεις, οι οποίες αντιστοιχούν και στα τρία διαφορετικά endpoints που ορίζει το auth route. Η πρώτη είναι η Register που διαχειρίζεται την εγγραφή του χρήστη. Ελέγχει την εγκυρότητα του username και του password που εισάγει ο χρήστης μέσω του request, δημιουργεί το κρυπτογραφημένο password και τελικά δημιουργεί ένα object του μοντέλου User με το username που έρχεται από το request και το αντίστοιχο κρυπτογραφημένο password, το οποίο και αποθηκεύει στη βάση δεδομένων. Η δεύτερη συνάρτηση είναι η Login, η οποία ψάχνει στη βάση δεδομένων για μια εγγραφή βάσει του username που έχει εισάγει ο χρήστης από το request, στη συνέχεια αν βρεθεί ο χρήστης, ελέγχει αν το password που έχει εισάγει ο χρήστης, ταιριάζει στο κρυπτογραφημένο password της εγγραφής που βρέθηκε στη βάση. Ο έλεγχος για την ομοιότητα των passwords γίνεται με τη βοήθεια της Verify μεθόδου της κλάσης BCrypt από το ομώνυμο πακέτο που αναλύθηκε νωρίτερα. Αν τα δύο αυτά βήματα ελέγχου ολοκληρωθούν με επιτυχία, τότε δημιουργείται ένα ζεύγος accessToken – refreshToken για τον χρήστη, διαγράφεται το προηγούμενο refreshToken του χρήστη ως μη έγκυρο πλεόν, αποθηκεύεται στη βάση δεδομένων το καινούριο refreshToken και τελικά αποστέλλεται σαν response το νέο ζεύγος από tokens. Η τρίτη συνάρτηση του AuthController είναι η RefreshToken. Αυτή η συνάρτηση περιμένει από το request το ζεύγος από tokens που διαθέτει ο χρήστης, με το accessToken να έχει λήξει. Γίνεται ένας αρχικός έλεγχος για το αν το request περιέχει αυτό το ζεύγος από tokens και στη συνέχεια γίνεται προσπάθεια εύρεσης του χρήστη από το accessToken με τη βοήθεια της συνάρτησης GetPrincipalFromExpiredToken από το IAuthService.

Αν οι παραπάνω έλεγχοι είναι επιτυχείς και βρεθεί ο χρήστης, τότε γίνεται αναζήτηση στη βάση για τα refesh tokens του χρήστη. Αν το refresh token του request αντιστοιχεί σε κάποιο έγκυρο refresh token του χρήστης από τη βάση τότε γίνεται διαγραφή των παλιών refresh tokens, πραγματοποιείται η δημιουργία ενός νέου ζεύγους access και refresh tokens και αποστέλλονται στο response. Σε διαφορετική περίπτωση επιστρέφεται μήνυμα σφάλματος και ο χρήστης αναγκαστηκά πρέπει να κάνει εκ νέου σύνδεση (login) στην εφαρμογή. Με αυτήν τη διαδικασία επιτυχάνεται η παράταση της σύνδεσης του χρήστη στην εφαρμογή με ασφαλή τρόπο, μιας και γίνεται η ανανέωση του access token του χωρίς να αναγκάζεται να κάνει login κάθε φορά που αυτό λήγει, ενώ παράλληλα, σε κάθε ανανέωση ανανεώνεται και το refresh token, το οποίο και διαθέτει μεγαλύτερη διάρκεια ζωής.

Ο δεύτερος controller που έχει υλοποιηθεί σε αυτό το Web API είναι ο TrackingController και αυτός διαθέτει μια συνάρτηση που αντιστοιχεί στο endpoint για την αποθήκευση δεδομένων. Η συνάρτηση αυτή πραγματοποιεί την δημιουργία ενός TrackingLog object με τα δεδομένα καταγραφής που έρχονται από το request και το αποθηκεύει στη βάση δεδομένων. Αυτός ο controller διαθέτει το attribute “[Authorize]” το οποίο υποδηλώνει ότι για τις λειτουργίες του, χρειάζεται ο χρήστης να είναι συνδεδεμένος στην εφαρμογή. Αυτό το attribute συνδέεται με την ύπαρξη ενός middleware ασφαλείας, το οποίο ελέγχει για το αν τα requests του χρήστη περιέχουν ένα έγκυρο access token (jwt) στο Authorization header. Το authorization middleware ορίζεται στο Program.cs αρχείο με τον κώδικα που φαίνεται στην παρακάτω εικόνα. Ο κώδικας αυτός χρησιμοποιεί το middleware από το πακέτο Microsoft.AspNetCore.Authentication.JwtBearer και ορίζει κάποιες επιλογές για την επικύρωση του access token του χρήστη.

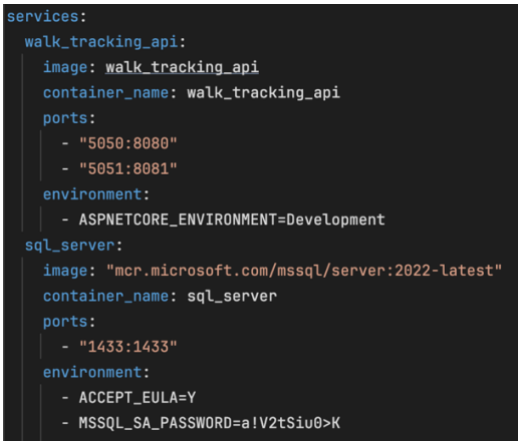
```
builder.Services.AddAuthentication(JwtBearerDefaults.AuthenticationScheme)
    .AddJwtBearer(options =>
    {
        options.TokenValidationParameters = new TokenValidationParameters
        {
            ValidateIssuerSigningKey = true,
            IssuerSigningKey = new SymmetricSecurityKey(
                Encoding.UTF8.GetBytes(builder.Configuration.GetSection(key: "AppSettings:Secret").Value!)),
            ValidateIssuer = false,
            ValidateAudience = false,
        };
    });
```

5.2.4 Η ενσωμάτωση του ελέγχου (validation) για το access token (jwt) στο Authentication middleware

Τελευταίο τμήμα της υλοποίησης του Web API και του backend συστήματος στο σύνολο, αφορά την υποστήριξη Docker. Για την υποστήριξη του docker και τη δημιουργία ενός container που μπορεί να τρέχει το Web API ήταν απαραίτητη η δημιουργία και υλοποίηση ενός Dockerfile αρχείου που θα περιέχει ουσιαστικά της οδηγίες για την δημιουργία του περιβάλλοντος εκτέλεσης της εφαρμογής εντός ενός docker container. Το Dockerfile που δημιουργήθηκε χωρίζει τη διαδικασία δημιουργίας της docker εικόνας σε τέσσερα στάδια. Στο πρώτο στάδιο γίνεται η λήψη της εικόνας του .net 8, η δήλωση του χρήστη της εφαρμογής, ο ορισμός του φακέλου που θα αντιγραφεί η εφαρμογή και ο ορισμός των θυρών (ports) από τις οποίες θα είναι προσβάσιμο το API. Στο δεύτερο στάδιο γίνεται η πραγματοποίηση του build της εφαρμογής, από το project, ενώ στο τρίτο στάδιο γίνεται και η εξαγωγή του build βιβλιοθήκη με τύπο αρχείου .dll. Τέλος, στο τέταρτο στάδιο γίνεται η έναρξη της εφαρμογής με χρήση του .NET cli.

Έτσι με τη χρήση αυτού του αρχείου μπορεί να γίνει η δημιουργία docker εικόνας για το Web API και μετέπειτα χρήση αυτής της εικόνας για την δημιουργία docker container που θα τρέχει το Web API. Υπάρχει βέβαια και η ανάγκη σύνδεσης του Web API με την δεδομένων, επομένως δημιουργήθηκε κι ένα docker-compose.yml αρχείο, για την συνδυασμένη εκτέλεση container τόσο για το Web API όσο και για τον Microsoft SQL Server ταυτόχρονα, σε ένα κοινό docker δίκτυο (network). Αυτό ίσως να μην είναι απαραίτητο για έναν υπολογιστή που έχει ως λειτουργικό σύστημα τα Windows της Microsoft, όμως στο μηχάνημα που πραγματοποιήθηκε η υλοποίηση και οι δοκιμές του συστήματος, δεν διατίθεται κάποια έκδοση του Sql Server για εγκατάσταση, ενώ αυτός ο τρόπος εκτέλεσης του Sql Server ίσως να ήταν βοηθητικός και σε κάποιον server όπου θα διέθετε λειτουργικό σύστημα Linux. Το αρχείο αυτό ορίζει σαν services το WebAPI ως walk_traking_api και τον Sql Server ως sql_server. Για καθένα από τα δύο αυτά services ορίζει τις εικόνες που πρέπει να χρησιμοποιή-σουν για τη δημιουργία containers. Η εικόνα για το Web API δημιουργείται από το Dockerfile που περιγράφηκε παραπάνω, ενώ η εικόνα του Sql Server λαμβάνεται από το DockerHub.

Επιπλέον, ορίζει port mappings ώστε να μπορούν τα services να είναι προσβάσιμα από συγκεκριμένα ports, καθώς και μεταβλητές περιβάλλοντος που περιλαμβάνουν διάφορες ρυθμίσεις, όπως το password για τον διαχειριστή της βάσης. Στην παρακάτω εικόνα φάινεται η δομή που έχει αυτό το αρχείο.



5.2.5 Η μορφή του αρχείου docker compose

Η εκτέλεση του backend με χρήση docker containers επιτυγχάνεται, από τον φάκελο του .NET project, με την εκτέλεση των εντολών:

- “docker build -t walk_tracking_api .”: για τη δημιουργία της εικόνας του Web Api
- “docker compose up -d”: για την εκτέλεση των containers του Backend στο σύνολο

Με τα παραπάνω ολοκληρώθηκε και η περιγραφή της λειτουργίας και της εκτέλεσης του backend συστήματος. Επόμενο, ζήτημα που θα αναλυθεί αποτελεί η δημιουργία δεδομένων για τους σκοπούς της εργασίας καθώς και η υλοποίηση του μοντέλου μηχανικής μάθησης με Tensorflow.

5.3 Δημιουργία Δεδομένων - Νευρωνικό Δίκτυο

Στόχος της συγκεκριμένης εργασίας, όπως περιεγράφηκε και στην αρχή του παρόντος εγγράφου είναι η υλοποίηση android εφαρμογής η οποία θα συλλέγει δεδομένα από τη συσκευή του χρήστη με τέτοιον τρόπο ώστε αυτά τα δεδομένα, με εφαρμογή κατάλληλων τεχνικών μηχανικής μάθησης και αξιοποίηση της τεχνολογίας του Tensorflow, να δημιουργηθεί ένα μοντέλο που να μπορεί να παράγει προτάσεις για τον αριθμό βημάτων που μπορεί να κάνει ένας χρήστης κατά την έναρξη της διαδρομής του. Για την δημιουργία και την εκπαίδευση ενός μοντέλου μηχανικής μάθησης, απαιτείται μεγάλος όγκος δεδομένων, ώστε το μοντέλο να μπορέσει να βρει τις συνδέσεις μεταξύ των δεδομένων, εάν υπάρχουν τέτοιες και να μπορεί να παράγει ακριβή αποτελέσματα. Για την εξυπηρέτηση των σκοπών της συγκεκριμένης εργασίας και με τον αναγκαίο όγκο δεδομένων που απαιτείται, πραγματοποιήθηκε η δημιουργία ενός δείγματος δεδομένων.

Ο λόγος που ακολουθήθηκε αυτή η προσέγγιση, της δημιουργίας μιας συλλογής δεδομένων, είναι ότι δεν βρέθηκε κάποια έτοιμη συλλογή δεδομένων που να παρέχει το είδος και τα χαρακτηριστικά τα οποία διαθέτουν τα δεδομένα που καταγράφονται από την εφαρμογή android, τα οποία κιόλας ερευνά η συγκεκριμένη εργασία. Παράλληλα, για την καταγραφή δεδομένων μέσω της εφαρμογής android που έχει υλοποιηθεί, θα χρειαζόταν ένας μεγάλος αριθμός δοκιμαστικών

εκτελέσεων ώστε να παραχθεί ο απαραίτητος όγκος δεδομένων, όπου κι εκεί προκύπτουν δύο νέα ζητήματα. Το πρώτο ζήτημα είναι ο χρόνος που απαιτείται για την δημιουργία της συλλογής δεδομένων μέσω δοκιμαστικών εκτελέσεων, μιας και κάθε δοκιμαστική διαδρομή μπορεί να διαρκέσει από μερικά λεπτά έως και μερικές ώρες. Το δεύτερο ζήτημα είναι η ομοιομορφία, υπό την έννοια ότι οι δοκιμαστικές εκτελέσεις που μπορεί να πραγματοποιήσει ένας συγκεκριμένος άνθρωπος, μπορεί να αποτυπώνουν στα δεδομένα και συγκεκριμένα (και μόνο) μοτίβα, ανάλογα με τις συνήθειες του και τις προτιμήσεις του. Μεγαλύτερης έκτασης ανάλυση όσον αφορά στο είδος και τη δημιουργία των δεδομένων θα πραγματοποιηθεί στις επόμενες ενότητες.

Συνεχίζοντας με τη διαδικασία δημιουργίας των δεδομένων, πρέπει να αναφερθεί ότι αυτά δημιουργήθηκαν με τη βοήθεια της γλώσσας Python και τη χρήση κάποιων βιβλιοθηκών όπως οι numpy και pandas την επεξεργασία τους σε μορφή πινάκων. Συγκεκριμένα υλοποίηθηκε ένα script σε python που συνδυάζει κάποιους περιορισμούς και απαιτήσεις για τη ενός δείγματος δημιουργία των δεδομένων. Αυτοί οι περιορισμοί και οι απαιτήσεις που εφαρμόστηκαν βάση του σχεδιασμού της εφαρμογής είναι το επόμενο ζήτημα που θα αναλυθεί.

Ξεκινώντας με τους περιορισμούς και τις απαιτήσεις που εφαρμόστηκαν στη δημιουργία δεδομένων, το πρώτο βήμα ήταν η επιλογή τοποθεσίας. Η εφαρμογή έχει ως στόχο την καταγραφή δεδομένων, ώστε να παράξει κάποιο συμπέρασμα για τον αριθμό των βημάτων που κάνουν οι χρήστες στην Αθήνα, σε συνδυασμό με τον καιρό και την ώρα. Επομένως οι τοποθεσίες που εμφανίζονται στα δεδομένα, έχουν περιοριστεί σε περιοχές εντός της Αθήνας. Συγκεκριμένα έχουν επιλεγεί κάποιες περιοχές της Αθήνας ως απόλυτες τοποθεσίες, με συγκεκριμένες τιμές για γεωγραφικό μήκος και γεωγραφικό πλάτος (longitude, latitude), γύρω από τις οποίες δημιουργούνται οι τοποθεσίες που υπάρχουν στα δεδομένα, με τυχαίο τρόπο. Συγκεκριμένα, κάθε διαφορετική τοποθεσία δημιουργεί μια συστάδα τοποθεσιών γύρω της. Με αυτόν τον τρόπο δημιουργείται ένα μοτίβο τοποθεσιών, κοντινό στην πραγματικότητα όπου οι άνθρωποι περπατούν σε συγκεκριμένες περιοχές που πιθανότατα είναι κατοικημένες και όχι σε περιοχές είναι δύσβατες λόγω του περιβάλλοντος χώρου, είτε αυτός είναι φυσικός, είτε αστικός (ποτάμια, απότομες πλαγιές, κτήρια λεωφόροι και άλλα).

Επόμενο σαν ζήτημα για τη δημιουργία δεδομένων είναι η επιλογή των καιρικών συνθηκών. Εδώ έπρεπε να συνδυαστεί η ετικέτα των καιρικών συνθηκών (“clear sky”, “rain”, “few clouds”, κτλ) μαζί με τη θερμοκρασία, ώστε να παραχθούν κάποιες λογικές τιμές στην ίδια εγγραφή ως προς αυτές. Για παράδειγμα σε μια εγγραφή, η ετικέτα του καιρού δεν μπορεί να είναι “snow” και ταυτόχρονα η θερμοκρασία να είναι 30 βαθμοί κελσίου, διότι φυσικά είναι αδύνατον. Επιπλέον οι καιρικές συνθήκες έπρεπε να προσαρμόζονται στην εποχή και στις συνήθεις συνθήκες της χώρας. Για παράδειγμα είναι αδύνατον, μια ημέρα μέσα στο καλοκαίρι να υπάρχει εγγραφή με θερμοκρασία στους μηδέν βαθμούς κελσίου. Επίσης στην Ελλάδα, ακόμα και τις εποχές εκτός καλοκαιριού οι θερμοκρασίες είναι πιο ψηλές από αυτές άλλων χωρών, ενώ επικρατεί ηλιοφάνεια το μεγαλύτερο μέρος του χρόνου. Οι ετικέτες των καιρικών συνθηκών που προέρχονται από το OpenWeatherMaps API αποτελούν ένα ευρύ σύνολο διαφορετικών ετικετών, οι οποίες για λόγους ευκολίας ομαδοποιήθηκαν σε ένα μικρότερο σύνολο, εκτός από τις πιο κοινά εμφανιζόμενες. Για παράδειγμα για την περίπτωση τις βροχής η ετικέτα “rain” έχει αρκετές διαφοροποιήσεις όπως “light rain”, “moderate rain”, “heavy intensity rain”, “heavy rain”, “very heavy rain”, “extreme rain” και άλλες πολλές ακόμα. Επομένως, παραμένοντας στο παράδειγμα για την περίπτωση της βροχής, διατηρήθηκαν τρεις ετικέτες, οι “light rain”, “shower rain” και όλες οι υπόλοιπες ομαδοποιήθηκαν κάτω από την ετικέτα “rain”. Η ίδια λογική ακολουθήθηκε και για τις υπόλοιπες ετικέτες καιρικών συνθηκών. Έπειτα, για να γίνουν πιο ρεαλιστικά τα δεδομένα, πραγματοποιήθηκε η δημιουργία ενός map με τις νέες (ομαδοποιημένες) ετικέτες καιρικών συνθηκών ως κλειδιά και οι αντίστοιχες τιμές τους, είναι οι πιθανότητες εμφάνισης αυτών των ετικετών, βάση ιστορικών δεδομένων καιρού εντός ενός έτους για την περιοχή της Αθήνας. Έτσι, για κάθε εγγραφή

που δημιουργήθηκε επιλέχθηκε με τυχαίο τρόπο η ετικέτα, βάσει και της πιθανότητας εμφάνισής της, με τη χρήση της συνάρτησης choice() από την numpy.random.

Παραμένοντας στο ζήτημα της επιλογής των καιρικών συνθηκών και διευρύνοντας το ένα βήμα ακόμα, κατά την επιλογή ημερομηνίας για τη δημιουργία timestamp, έπρεπε να ληφθεί υπ'όψην το γεγονός ότι σε διαφορετικές εγγραφές της ίδιας ημέρας και σε συνδυασμό με την ώρα, πρέπει οι καιρικές συνθήκες να είναι παρόμοιες. Για παράδειγμα για την ίδια ημέρα, με την ίδια ώρα ή και με μερικές ώρες διαφορά, σε δυο διαφορετικές εγγραφές, δεν είναι πολύ πιθανό να υπάρχει διαφορά θερμοκρασίας της τάξεως των δέκα βαθμών κελσίου, μεταξύ των δύο εγγραφών. Συνεπώς έχει ληφθεί υπ' όψην και αυτή η παράμετρος, ώστε οι εγγραφές της ίδιας ημερομηνίας, με παρόμοια ώρα στο timestamp να έχουν παρόμοιες θερμοκρασίες. Αυτό τεχνικά επιτεύχθηκε με την δημιουργία ενός dictionairy, στο οποίο αποθηκευόταν ο συνδυασμός κάθε νέας ημερομηνίας και ώρας μαζί με την επιλεγμένη θερμοκρασία. Με αυτόν τον τρόπο κατά την δημιουργία ενός νέου συνδυασμού ημερομηνίας και ώρας στη δημιουργία μιας νέας εγγραφής, γινόταν αναζήτηση στο dictionairy για την ύπαρξη της νέας ημερομηνίας, ώστε αν υπάρχει, να ανακτηθεί η υπάρχουσα τιμή θερμοκρασίας και η θερμοκρασία της νέας εγγραφής, να δημιουργηθεί τυχαία μέσα από ένα μικρό εύρος τιμών που περιβάλλουν την υπάρχουσα θερμοκρασία. Συνεπώς, στο σύνολο των εγγραφών δεν θα προκύπτουν εγγραφές με κοινά χαρακτηριστικά, τα οποία εμφανίζουν μη λογικές διαφοροποιήσεις, βοηθώντας στο τελικό αποτέλεσμα.

Εφόσον, έγινε η αναφορά στην επιλογή της ημερομηνίας και της ώρας για κάθε εγγραφή, σε αυτό το σημείο θα γίνει και η ανάλυση των περιορισμών αλλά και της υλοποίησης της επιλογής αυτών. Ο περιορισμός που τέθηκε όσον αφορά την ημερομηνία, είναι ότι στο σύνολο των δεδομένων εντάχθηκαν ημερομηνίες που αφορούν το έτος 2024 και οι οποίες δημιουργήθηκαν με τυχαίο τρόπο ώστε να καταλαμβάνουν διάφορες ημέρες και μήνες μέσα στο χρόνο. Εδώ άλλος ένας περιορισμός που τέθηκε είναι αυτός της θερμοκρασίας ανά εποχή. Για παράδειγμα, είναι αδύνατον το χειμώνα να υπάρχει κάποια ημέρα με 40 βαθμούς κελσίου. Έτσι δημιουργήθηκαν κάποια εύρη θερμοκρασιών ανά εποχή. Κατόπιν της δημιουργίας της ημερομηνίας, διεξάγεται ένας έλεγχος για τον μήνα που έχει επιλεγεί, από την ημερομηνία και βάση του μήνα προκύπτει και το αντίστοιχο εύρος τιμής για τις θερμοκρασίες. Έπειτα από το επιλεγμένο εύρος θερμοκρασιών, επιλέγεται με τυχαίο τρόπο μια τιμή θερμοκρασίας.

Όσον αφορά την επιλογή της τιμής για την ώρα που αναπαριστά την ώρα έναρξης μιας διαδρομής, η επιλογή έγινε από ένα εύρος τιμών και πάλι. Αυτό το εύρος τιμών είναι περιορίζει τις ώρες έναρξης διαδρομών από τις 8 προ μεσημβρίας έως τις 11 προ μεσημβρίας. Η επιλογή του περιορισμού των ωρών μέσα από αυτό το εύρος, προέκυψε από την ιδέα, ότι οι άνθρωποι συνήθως περπατούν ή πραγματοποιούν κάποια δραστηριότητα κυρίως σε ώρες εντός αυτού του συνόλου ωρών. Είναι δύσκολο ένας άνθρωπος να πραγματοποιεί μια αξιοσημείωτη από άποψη χρονικής διάρκειας, διαδρομή ή δραστηριότητα την οποία είναι διατεθειμένος και να καταγράψει, στις τρεις τα ξημερώματα, για παράδειγμα. Παράλληλα, αν ωστόσο το πράξει αυτό κάποιος, η εγγραφή που θα δημιουργηθεί θα αποτελεί μια μειονότητα για τα δεδομένα, που ενδεχομένως να αποτελεί και «ακραία» τιμή εισόδου για το μοντέλο που θα δημιουργηθεί, μιας και ο περισσότερος κόσμος περπατά ή πραγματοποιεί κάποια φυσική δραστηριότητα τις πρωινές, τις απογευματινές ώρες μέχρι νωρίς το βράδυ.

Ακόμη μερικοί περιορισμοί που τέθηκαν στη δημιουργία δεδομένων, αφορούν την επιλογή της φωτεινότητας της οθόνης. Η φωτεινότητα της οθόνης αποτελεί ένα εξατομικευμένο δεδομένο μιας κι ο κάθε χρήστης ορίζει τη φωτεινότητα της οθόνης στα επίπεδα που επιθυμεί ο ίδιος, βάσει τον αναγκών και των συνηθειών του. Στις android συσκευές ωστόσο, η φωτεινότητα συνήθως ρυθμίζεται αυτόματα, βάσει των επιπέδων φωτισμού που παρέχει το περιβάλλον του χρήστη.

Οι τιμές που αποδίδει το service που επιστρέφει την φωτεινότητα μιας συσκευής από την εφαρμογή android, κυμαίνονται από 0 (εντελώς κλειστή οθόνη) έως 255 που είναι η τιμή της μέγιστης φωτεινότητας. Από τα δεδομένα που διαθέτει το μοντέλο της συγκεκριμένης εργασίας, δύο είναι αυτά που θεωρητικά επηρεάζουν τα επίπεδα φωτεινότητας. Το πρώτο είναι η ώρα μέσα στην ημέρα, η οποία επηρεάζει το επίπεδο φωτεινότητας της οθόνης, αφού κατά τη διάρκεια της ημέρας αλλάζει και ο φωτισμός στο φυσικό περιβάλλον. Έτσι τις πρωινές ώρες το επίπεδο φωτεινότητας μιας συσκευής τείνει να είναι υψηλότερο, μιας και ο ήλιος βρίσκεται σε ψηλή θέση και φωτίζει το περιβάλλον, ενώ τις απογευματινές και τις βραδινές ώρες που πέφτει ο ήλιος τείνει να χαμηλώνει και η φωτεινότητα των συσκευών, ώστε να μην κουράζει και επιβαρύνει την όραση του χρήστη. Το δεύτερο δεδομένο που επηρεάζει τη φωτεινότητα μιας συσκευής, με έμμεσο τρόπο βέβαια, είναι η ετικέτα καιρικών συνθηκών. Όταν έχει συννεφιά ή βρέχει και γενικότερα σε περιπτώσεις που οι καιρικές συνθήκες καθιστούν το φωτισμό στο περιβάλλον χαμηλότερο, προκαλούν και τη μείωση τις φωτεινότητας στις συσκευές. Για τους λόγους αυτούς, έχουν οριστεί κάποια εύρη φωτεινότητας από τα οποία επιλέγεται και η τιμή της φωτεινότητας, με τυχαίο τρόπο, μέσω ελέγχου που εξετάζει την τιμή της ώρας μέσα στην ημέρα, που έχει επιλεγεί για την εγγραφή προς δημιουργία, συνδυαστικά με την ετικέτα των καιρικών συνθηκών.

Τέλος, για τη δημιουργία μιας εγγραφής, χρειάζεται και μια τιμή για τον αριθμό των βημάτων. Η τιμή αυτή δεν μπορεί να περιοριστεί αυστηρά μιας και δεν υπάρχει κάποιο όριο στο πόσα βήματα πραγματοποιεί κάποιος, τόσο προς τα πάνω όσο και προς τα κάτω. Κάθε άνθρωπος διαθέτει τις δικές του συνήθειες, αντοχές, ανάγκες και διαθέσιμο χρόνο για να πραγματοποιήσει έναν αριθμό βημάτων. Συνεπώς εδώ έχουν εφαρμοστεί δύο περιορισμοί διαισθητικά, εμπειρικά και βάση αποτελεσμάτων από έρευνες. Ο πρώτος περιορισμός αφορά τα όρια πλήθους βημάτων που εκτελούν οι χρήστες κι εδώ έχουν τεθεί ως τέτοια για το κατώτερο όριο τα 1.000 βήματα, ενώ για το ανώτερο όριο τα 16.000 βήματα. Ο δεύτερος περιορισμός αφορά μια εκτίμηση για το πλήθος βημάτων βάση θερμοκρασίας, θεωρώντας ότι σε πολύ υψηλές ή πολύ χαμηλές θερμοκρασίες, τα πλήθη βημάτων τείνουν να είναι χαμηλότερα από ότι σε μεσαίες θερμοκρασίες, δηλαδή σε συνθήκες που ένας άνθρωπος νιώθει πιο άνετος να κάνει περισσότερα βήματα. Επιπλέον έγινε η παραδοχή ότι οι καιρικές συνθήκες, δεν είναι καθοριστικός παράγοντας για έναν άνθρωπο ώστε να περπατήσει ένα μεγαλύτερο ή μικρότερο πλήθος βημάτων από αυτό που συνηθίζει. Για παράδειγμα στις περιπτώσεις που υπάρχει συννεφιά, λογικά ένας άνθρωπος μπορεί να πετύχει το ίδιο πλήθος βημάτων σε σύγκριση με τις περιπτώσεις που ο ουρανός είναι καθαρός, ενώ αν βρέχει, ή αν υπάρχει χιονόπτωση ή χαλαζόπτωση, το πιθανότερο είναι ότι δεν θα περπατήσει καθόλου, και αν πράξει το αντίθετο, θα είναι για λόγους ανάγκης, που σε αυτήν την περίπτωση ενδεχομένως, δε θα είναι διατεθειμένος και να καταγράψει τη διαδρομή του.

Έχοντας θέσει τους παραπάνω περιορισμούς σε μορφή κώδικα για την δημιουργία κάθε εγγραφής, προστέθηκε και ένα userId σε κάθε εγγραφή με τιμές τυχαίες μεταξύ 1 και 10 (δέκα χρήστες) ώστε τα δεδομένα των εγγραφών τα είναι πανομοιότυπα σε μορφή με αυτά που αποθηκεύονται στη βάση δεδομένων μέσω της εφαρμογής. Δημιουργήθηκαν 10.000 εγγραφές οιοποίες εξάχθηκαν σε ένα αρχείο σε μορφή csv.

Το επόμενο βήμα προς υλοποίηση ήταν η φόρτωση αυτού του αρχείου, η εξαγωγή δεδομένων και προεπεξεργασία τους, ώστε μετά να ακολουθήσει η δημιουργία του μοντέλου Tensorflow. Για την προεπεξεργασία των δεδομένων και τη δημιουργία του μοντέλου μηχανικής μάθησης έγινε εγκατάσταση και εισαγωγή των παρακάτω βιβλιοθηκών:

- Numpy
- Pandas
- Scikit Learn (sklearn)
- Tensorflow

Η βιβλιοθήκη numpy χρησιμοποιείται για επιστημονικούς υπολογισμούς, για αριθμητικούς υπολογισμούς, καθώς και για χειρισμό και πράξεις πολυδιάστατων πινάκων. Η βιβλιοθήκη pandas χρησιμοποιείται για την ανάλυση και διαχείριση δεδομένων. Αυτή η βιβλιοθήκη είναι ιδιαίτερα δημοφιλής στον τομέα της επιστήμης δεδομένων μιας και παρέχει εργαλεία για τη διαχείριση δομημένων δεδομένων όπως πίνακες και χρονικές σειρές. Η πιο δημοφιλής δομή που διαθέτει είναι το DataFrame το οποίο μπορεί να εισάγει και να αναπαριστά δισδιάστατα δεδομένα με σειρές και στήλες (όπως αρχεία csv, xlsx και πίνακες βάσεων δεδομένων SQL). Η βιβλιοθήκη Scikit-learn (sklearn) είναι επίσης μια δημοφιλής βιβλιοθήκη σε γλώσσα Python που χρησιμοποιείται για την υλοποίηση μοντέλων μηχανικής μάθησης, αφού διαθέτει ένα μεγάλο σύνολο από εργαλεία για ανάλυση δεδομένων και αλγόριθμους μηχανικής μάθησης. Τα εργαλεία που διαθέτει για την ανάλυση δεδομένων είναι ιδανικά και για την προεπεξεργασία τους σε συνδυασμό με τη χρήση της βιβλιοθήκης pandas. Τέλος η βιβλιοθήκη Tensorflow, όπως περιεγράφηκε και νωρίτερα είναι μια πολύ δημοφιλής βιβλιοθήκη που δημιουργήθηκε από την Google και παρέχει τα κατάλληλα εργαλεία για τη δημιουργία και εκπαίδευση μοντέλων μηχανικής μάθησης και βαθιάς μάθησης.

Επομένως, σε ένα καινούριο αρχείο python έγινε εισαγωγή των παραπάνω βιβλιοθηκών κι έπειτα ακολούθησε η εισαγωγή των δεδομένων που δημιουργήθηκαν, από το csv αρχείο. Η εισαγωγή των δεδομένων έγινε με τη χρήση της μεθόδου read_csv() της βιβλιοθήκης pandas σε μια μεταβλητή με όνομα data. Η μέθοδος read_csv επιστρέφει τα δεδομένα που περιέχονται στο csv σε ένα Dataframe. Το DataFrame αποτελεί την ιδανική δομή για την αναπαράσταση των δεδομένων μιας κι παρέχει μεγάλη ευκολία στον χειρισμό των δεδομένων σε στήλες. Κάθε στήλη του DataFrame που έχει δημιουργηθεί, ουσιαστικά αναπαριστά και καθένα από τα χαρακτηριστικά που θα αποτελέσουν την είσοδο του μοντέλου μηχανικής μάθησης, ή μια πηγή για τη δημιουργία νέων χαρακτηριστικών που τελικά θα αποτελέσουν μέρος της εισόδου το μοντέλου.

Ξεκινώντας με την προεπεξεργασία των δεδομένων, το πρώτο βήμα που επιτελέστηκε ήταν η μετατροπή των κατηγορικών των δεδομένων σε αριθμητικές τιμές. Από τα χαρακτηριστικά των δεδομένων που έχουν δημιουργηθεί και αποτελούν προφανώς και χαρακτηριστικά των δεδομένων καταγραφής, η ετικέτα των καιρικών συνθηκών αποτελεί κατηγορικό δεδομένο. Η ετικέτα των καιρικών συνθηκών ως χαρακτηριστικό που θα εισαχθεί στο μοντέλο μηχανικής μάθησης, πρέπει να μετατραπεί σε αριθμητικό δεδομένο, μιας και τα μοντέλα μηχανικής και βαθιάς μάθησης μπορούν να έχουν ως είσοδο μόνο αριθμητικά δεδομένα. Η επεξεργασία, τροποποίηση και μετασχηματισμός των δεδομένων σε χαρακτηριστικά εισόδου για ένα μοντέλο, σαν διαδικασία συνολικά ονομάζεται Μηχανική Χαρακτηριστικών (Feature Engineering). Συνεχίζοντας, για την ετικέτα καιρικών συνθηκών, ως στήλη του DataFrame μετατράπηκε σε τύπο Category, μιας και περιέχει πεπερασμένο αριθμό διαφορετικών τιμών, όπως αναλύθηκε και παραπάνω. Δηλαδή κάθε διακριτή τιμή που περιέχει αυτή η στήλη αποτελεί μια κατηγορία. Έπειτα η τιμές της στήλης μετατράπηκαν σε αριθμητικές τιμές ως οι αριθμοί που αντιστοιχούν στην κάθε κατηγορία. Με αυτόν τον τρόπο το μόνο μη αριθμητικό δεδομένο που διαθέτει το DataFrame με τα δεδομένα, μετατράπηκε σε αριθμητικό.

Το επόμενο δεδομένο που χρειάστηκε επεξεργασία ήταν το timestamp. Το timestamp σε milliseconds, ως δεδομένο είναι τύπου «μεγάλου» ακεραίου αριθμού (στο Android είναι τύπου Long).Επιπλέον αναπαριστά χρονικές στιγμές, ως ακριβής τιμή για μια συγκεκριμένη ημερομηνία, συγκεκριμένη ώρα, λεπτά, δευτερόλεπτα και χιλιοστά του δευτερολέπτου. Αυτό πρακτικά μεταφράζεται στο γεγονός ότι μπορεί να λάβει σχεδόν αμέτρητες τιμές, με μεγάλες διακυμάνσεις μεταξύ τους. Αυτό σημαίνει ότι από μόνο του δεν είναι ιδιαίτερα χρήσιμο από μόνο του για χρήση ως χαρακτηριστικό εισόδου ενός μοντέλου. Ωστόσο, όμως περιέχει αρκετή πληροφορία η οποία και μπορεί να εξαχθεί αλλά και να είναι κα χρήσιμη ως χαρακτηριστικό εισόδου. Συνεπώς από το timestamp προέκυψαν δύο νέες στήλες που αποτέλεσαν χαρακτηριστικά εισόδου στο μοντέλο. Με μετατροπή της στήλης σε τύπο datetime προέκυψαν οι στήλες ‘hour’ και ‘day_of_week’.

Η στήλη hour αναπαριστά την «ώρα της ημέρας» λαμβάνοντας τιμές από που προέρχεται από το timestamp, λαμβάνοντας τιμές από 0 έως 23, για κάθε μια ώρα στη διάρκεια μιας μέρας, ενώ η στήλη day_of_week αναπαριστά την ημέρα της εβδομάδας και λαμβάνει τιμές από 0 (που αντιστοιχεί στην Δευτέρα) έως 6 (που αναπαριστά την Κυριακή).

Διατηρώντας τις υπόλοιπες τιμές όπως ως έχουν, η είσοδος που προκύπτει, αποτελείται από τα χαρακτηριστικά 'latitude', 'longitude', 'temperature', 'weatherLabel', 'brightness', 'hour' και 'day_of_week'. Ωστόσο κάποια χαρακτηριστικά έχουν μεγάλα εύρη τιμών, με μεγάλη διακύμανση μεταξύ των τιμών τους, όπως τα latitude, longitude, temperature και brightness, ενώ οι κλίμακες των διαφορετικών χαρακτηριστικών είναι πολύ διαφορετικές μεταξύ τους. Αυτό σημαίνει ότι χαρακτηριστικά όπως τα latitude, longitude ή το brightness που διαθέτουν μεγάλη κλίμακα τιμών, μπορεί να επηρεάσουν τους υπολογισμούς των βαρών κατά την εκπαίδευση του μοντέλου. Γι’ αυτόν το λόγο πραγματοποιήθηκε κανονικοποίηση της εισόδου, ώστε να εξασφαλιστεί ότι τα δεδομένα θα έχουν παρόμοια κλίμακα και δεν θα επηρεάζουν δυσανάλογα το αποτέλεσμα. Επιπλέον, η κανονικοποιήση βοηθά στην βελτίωση της σύγκλισης του μοντέλου, δηλαδή οδηγεί σε γρηγορότερη και πιο σταθερή εκπαίδευση. Για την κανονικοποίηση της εισόδου χρησιμοποιήθηκε ο MinMaxScaler από την βιβλιοθήκη Scikit-learn. Ο MinMaxScaler φέρνει τις κλίμακες των χαρακτηριστικών σε τιμές από 0 έως 1 και είναι κατάλληλος για νευρωνικά δίκτυα.

Ολοκληρώνοντας με τον μετασχηματισμό της εισόδου, ως έξοδος ορίστηκε η στήλη ‘steps’ η οποία αναπαριστά τα βήματα που πραγματοποιήθηκαν σε κάθε εγγραφή και αποτελεί και την τιμή προς πρόβλεψη από το μοντέλο που θα δημιουργηθεί. Κατόπιν του ορισμού και της εξόδου, πραγματοποιήθηκε ο διαχωρισμός της εισόδου και της εξόδου, σε ένα σετ για εκπαίδευση του μοντέλου κι ένα για δοκιμές, για καθεμία αντίστοιχα. Το 20% του δείγματος ορίστηκε σαν δείγμα δοκιμών και το υπόλοιπο 80% αποτελεί το δείγμα εκπαίδευσης του μοντέλου.

Στη συνέχεια δημιουργήθηκε το νευρωνικό δίκτυο που θα κάνει προβλέψεις για τον αριθμό βημάτων που δύναται να κάνει ένας χρήστης, βάσει των παρατηρήσεων (χαρακτηριστικών) που προκύπτουν από την εφαρμογή android. Πρόκειται για ένα Sequential μοντέλο, που προέρχεται από τη βιβλιοθήκη Keras του Tensorflow. Το Sequential μοντέλο χρησιμοποιείται για τη δημιουργία ενός σειριακού νευρωνικού δικτύου, δηλαδή ενός νευρωνικού δικτύου που αποτελείται από επίπεδα (layers) που τοποθετούνται, το ένα μετά το άλλο σε μια ευθεία ακολουθία. Χρησιμοποιείται όταν τα δεδομένα ρέουν γραμμικά από το ένα επίπεδο στο επόμενο, δηλαδή από το επίπεδο εισόδου τα δεδομένα ρέουν στο πρώτο κρυφό επίπεδο και στη συνέχεια, η έξοδος το πρώτου επιπέδου γίνεται η είσοδος του επόμενου, μέχρι τα δεδομένα να καταλήξουν στο επίπεδο εξόδου και να παραχθεί η τελική έξοδος. Αυτό το μοντέλο είναι ιδανικό για γρήγορη δημιουργία και εκπαίδευση απλών μοντέλων.

Το πρώτο επίπεδο που δηλώνεται στο μοντέλο και πρακτικά αποτελεί το επίπεδο εισόδου είναι ένα InputLayer που παίρνει ως παράμετρο την διάσταση των χαρακτηριστικών εισόδου. Το μοντέλο αποτελείται από ακόμα τρία Dense επίπεδα με 64, 32 και 16 νευρώνες αντίστοιχα, που είναι τα κρυφά επίπεδα και ένα ακόμα Dense επίπεδο που αποτελεί το επίπεδο εξόδου, προφανώς με έναν νευρώνα ο οποίος παράγει μια, συνεχή τιμή και αυτή είναι ο προβλεπόμενος αριθμός βημάτων. Το Dense επίπεδο είναι ένα πλήρως συνδεδεμένο επίπεδο, δηλαδή κάθε νευρώνας του συνδέεται με όλους τους νευρώνες του προηγούμενου επιπέδου. Είναι ιδανικό για προβλήματα παλινδρόμησης, όπως ως τέτοιο εξετάστηκε το τρέχον και για προβλήματα ταξινόμησης. Κάθε Dense επίπεδο, πέραν του αριθμού των νευρώνων, παίρνει επίσης σαν παράμετρο μια συνάρτηση ενεργοποίησης. Η συνάρτηση ενεργοποίησης που χρησιμοποιήθηκε ήταν η ReLU (rectified linear activation function), η οποία είναι μια γραμμική συνάρτηση που βγάζει σαν έξοδο την είσοδο αν αυτή είναι θετική και βγάζει σαν έξοδο 0 αν η είσοδος είναι αρνητική. Αυτή η συνάρτηση έχει γίνει η προκαθορισμένη σε πολλούς τύπους νευρωνικών δικτύων, διότι για τα μοντέλα τα οποία την χρησιμοποιούν εκπαιδεύονται

ευκολότερα, για παράδειγμα η εκπαίδευση είναι πιο γρήγορη απ'ότι με τη χρήση Σιγμοειδούς συνάρτησης και συχνά πετυχαίνουν και καλύτερη απόδοση.

Μεταξύ των τριών κρυφών επιπέδων έχουν τοποθετηθεί και τέσσερα ακόμα επίπεδα, ανα δύο ίδια μεταξύ τους, δύο μεταξύ του πρώτου κρυφού επιπέδου και του δεύτερου και δύο μεταξύ του δεύτερου και του τρίτου. Αυτά τα επιπλεόν επίπεδα είναι για κάθε ζεύγος, ένα BatchNormalization κι ένα Dropout επίπεδο. Αυτά τα επίπεδα δεν είναι είναι επίπεδα υπολογισμού, αλλά επιτελούν κάποιες σημαντικές εργασίες για την βελτίωση της απόδοσης του μοντέλου. Συγκεκριμένα το BatchNormalization επίπεδο κανονικοποιεί τις εξόδους του επιπέδου που προηγείται, ώστε αυτές να έχουν μέση τιμή 0 και διακύμανση 1. Πρακτικά επιταχύνει και σταθεροποιεί την εκπαίδευση του μοντέλου. Από την άλλη το Dropout επίπεδο, λαμβάνοντας μια αριθμητική τιμή σαν παράμετρο, ανάλογα με την τιμή της παραμέτρου που χρησιμοποιεί σαν ποσοστό, επιλέγει αυτό το ποσοστό των εξόδων του προηγούμενου επιπέδου προς αποκλεισμό. Οι αποκλεισμένοι νευρώνες απενεργοποιούνται από τους υπολογισμούς. Η χρήση αυτών των δύο επιπέδων συνδυαστικά βοηθά στο να μην υπερεκπαιδεύεται το μοντέλο, δηλαδή στο να μπορεί να γενικευτεί περισσότερο, ώστε σε διαφορετικά δεδομένα από αυτά του δείγματος, να διατηρεί την ίδια υψηλή απόδοση.

Με την ολοκλήρωση του καθορισμού του μοντέλου, έγινε η κλήση της compile μεθόδου του, η οποία καθορίζει τις παραμέτρους εκπαίδευσης του μοντέλου. Αυτή η μέθοδος παίρνει κάποιες παραμέτρους, όπως βελτιστοποιητές, συναρτήσεις κόστους και μετρικές. Εδώ έχει χρησιμοποιηθεί ο βελτιστοποιητής Adam (Adaptive Moment Estimation) με ρυθμό εκμάθησης 0.01. Ο Adam είναι ένας αλγόριθμος που συνδυάζει τα πλεονεκτήματα των "Stochastic Gradient Descent with momentum" και "RMSProp" αλγορίθμων. Χρησιμοποιεί τις τετραγωνικές κλίσεις για την κλιμάκωση του ρυθμού μάθησης, όπως ο RMSprop, και εκμεταλλεύεται το momentum χρησιμοποιώντας τον κινητό μέσο όρο της κλίσης αντί της ίδιας της κλίσης, όπως ο SGD με momentum. Με αυτόν τον τρόπο συνδυάζει τον Δυναμικό Ρυθμό Μάθησης και την Εξομάλυνση για να φτάσει στα γενικά ελάχιστα. Ο ρυθμός εκμάθησης καθορίζει το πόσο γρήγορα το μοντέλο προσαρμόζει τα βάρη του. Μικρότερος αριθμός για τον ρυθμό εκμάθησης κάνει την εκπαίδευση πιο αργή, αλλά ταυτόχρονα πιο σταθερή. Πέραν του βελτιστοποίηση, στην compile περνιούνται και οι παράμετροι mean_squared_error ως συνάρτηση κόστους και η 'mae' – Mean Absolute Error σαν μετρική για την εμφάνιση του σφάλματος μεταξύ της πρόβλεψης και της πραγματικής τιμής.

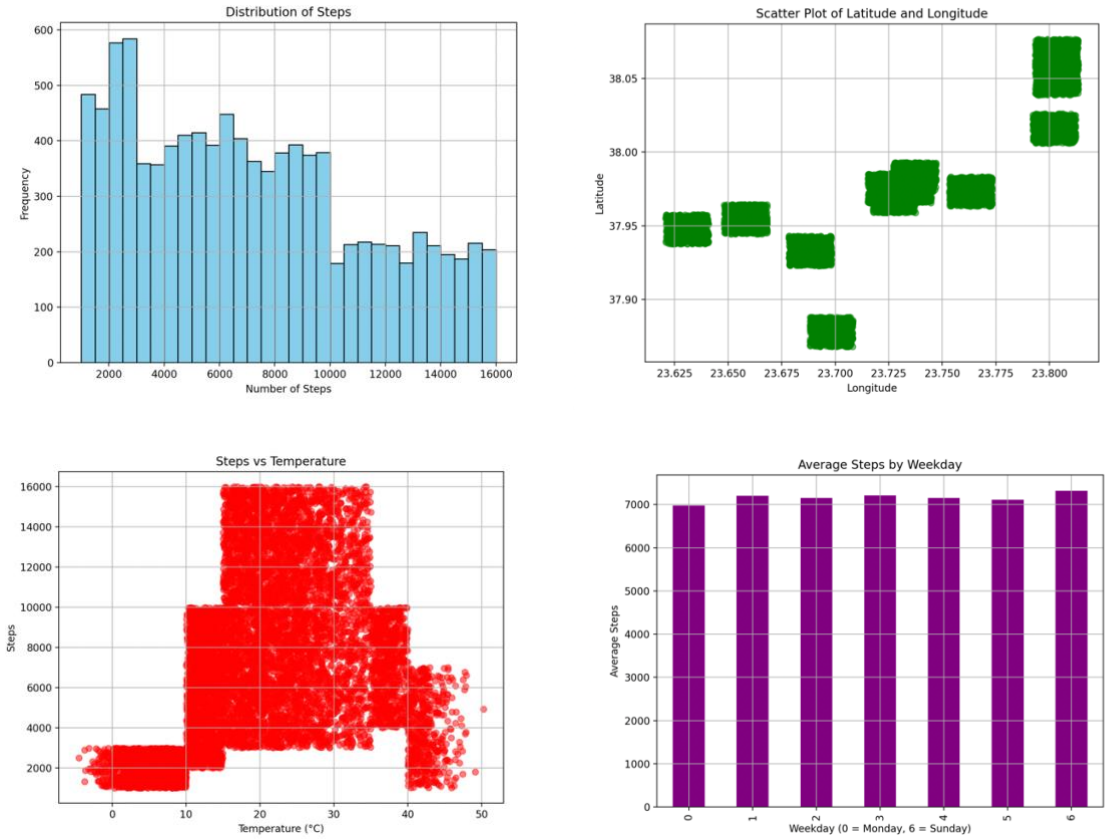
Στη συνέχεια, ακολούθησε η εκπαίδευση του μοντέλου με την κλήση της μεθόδου fit από το μοντέλο, με παραμέτρους, φυσικά, το διάνυσμα εισόδου, με έξοδο το διάνυσμα εξόδου, σαν δεδομένα επαλήθευσης το τμήμα της αρχικής συλλογής δεδομένων που χωρίστηκε για δοκιμές και επαλήθευση, όπως περιεγράφηκε νωρίτερα, τα epochs σε 100 εποχές, δηλαδή σε 100 επαναλήψεις εκπαίδευσης του μοντέλου με το σετ δεδομένων εκπαίδευσης, το batch_size σε μέγεθος 32, δηλαδή τα δεδομένα χωρίζονται σε παρτίδες των 32 δειγμάτων και τα βάρη του μοντέλου προσαρμόζονται μετά από κάθε παρτίδα που περνάει από το μοντέλο, με τελευταία παράμετρο, σαν callback, μια συνάρτηση για πρόωρο τερματισμό. Αυτό το callback πρόκειται για ένα. instance EarlyStopping που προέρχεται από το Keras και ουσιαστικά σταματάει την διαδικασία εκπαίδευσης του μοντέλου, σε περίπτωση που αυτό φτάσει στο μέγιστο δυνατό αποτέλεσμα, με σκοπό να αποφευχθεί η υπερεκπαίδευση στα δεδομένα εισόδου (overfitting). Το EarlyStopping παίρνει σαν παραμέτρους, την μετρική του μοντέλου που πρέπει να παρακολουθεί, που εδώ είναι το val_loss, δηλαδή η τιμή του σφάλματος, την παράμετρο patience που δηλώνει τον αριθμό εποχών στις οποίες δεν βελτιώνεται το σφάλμα πριν τερματίσει την εκπαίδευση, και τέλος την παράμετρο restore_best_weights η οποία είναι τύπου Boolean και δηλώνει το, αν κατά τον τερματισμό της εκπαίδευσης, είναι επιθυμητή η εξαγωγή των καλύτερων βαρών που έχουν παραχθεί μέχρι τη στιγμή πρόωρου τερματισμού.

Σε αυτό το σημείο εφόσον έχει ολοκληρωθεί και η δημιουργία του μοντέλου, έχει υλοποιηθεί και ένα τελευταίο τμήμα κώδικα, το οποίο, κατόπιν της εκπαίδευσης του μοντέλου, μετατρέπει το μοντέλο Keras, σε μοντέλο Tensorflow Lite κι έπειτα το εξάγει σε ένα αρχείο με τύπο ".tflite". Αυτό το αρχείο στη συνέχεια ενσωματώθηκε στον κώδικα της εφαρμογής Android και με κατάλληλους μετασχηματισμούς στα δεδομένα που συλλέγονται από την εφαρμογή, ώστε να έρχονται στην μορφή που απαιτεί η είσοδος του μοντέλου, όταν τη χρησιμοποιεί ο χρήστης, μπορεί να παράξει μια πρόβλεψη για τον αριθμό βημάτων που αυτός θα μπορούσε να διανύσει ξεκινώντας τη διαδρομή του. Γενικότερα η ιδέα είναι βάσει καινούριων παρατηρήσεων/δεδομένων που προκύπτουν από καταγραφές χρηστών, αυτές να εξάγονται από τη βάση και να τοποθετούνται στο μοντέλο για επανεπαίδευση και ενδεχόμενη βελτίωσή του όσον αφορά την απόδοση, όσο προκύπτουν διαφορετικά δεδομένα.

6. Συμπεράσματα

Από την ανάλυση των δεδομένων που δημιουργήθηκαν και κατά την εκπαίδευση του μοντέλου προέκυψαν μερικά συμπεράσματα. Το μοντέλο μπορεί να παράγει προβλέψεις για τα βήματα που θα μπορούσε να κάνει ένας χρήστης, με απόλυτο σφάλμα περίπου στα 2.500 βήματα. Όπως αναφέρεται βιβλιογραφικά σε διάφορες έρευνες, το αποδεκτό απόλυτο σφάλμα σε αριθμό βημάτων, συνήθως εξαρτάται από την χρήση και το ερευνητικό πεδίο. Για παράδειγμα, σε περιπτώσεις ιατρικών ερευνών αυτό πρέπει να είναι αρκετά χαμηλό, δηλαδή χαμηλότερο από 500 βήματα σαν απόλυτο σφάλμα. Ενώ σε πεδία φυσικών/αθλητικών δραστηριοτήτων ή άλλων γενικών περιπτώσεων συνήθως το απόλυτο σφάλμα σε αριθμό βημάτων που είναι αποδεκτό, κυμαίνεται από 300 έως 1000 βήματα. Δεδομένων αυτών των παρατηρήσεων η απόδοση του μοντέλου χρειάζεται βελτίωση για την παραγωγή ακριβέστερων προβλέψεων. Οι μέθοδοι και οι τεχνικές που έχουν χρησιμοποιηθεί για την δημιουργία του μοντέλου και περιγράφονται παραπάνω, παρέχουν τα καλύτερα αποτελέσματα από αυτές που έχουν δοκιμαστεί συνολικά.

Συνεπώς, προκύπτουν κάποιες υποθέσεις προς εξέταση για την ανάλυση των δεδομένων που χρησιμοποιήθηκαν για την δημιουργία του μοντέλου. Η πρώτη υπόθεση έχει να κάνει με την μορφή, την κατανομή και τα μοτίβα που σχηματίζουν τα δεδομένα που δημιουργήθηκαν. Σίγουρα η δημιουργία δεδομένων με χειροκίνητο τρόπο δεν μπορεί να οδηγήσει σε ένα αποτέλεσμα πολύ κοντά στο πραγματικό, αλλά να εφαρμόζει μόνο κάποια μοτίβα, όπως εφαρμόστηκαν με την προσθήκη περιορισμών. Η δημιουργία των δεδομένων έχει πραγματοποιηθεί βάση λογικών υποθέσεων, όμως μπορεί να μην λαμβάνονται όλες οι περιπτώσεις χαρακτηριστικών που απεικονίζονται στα δεδομένα. Επιπλέον, υπάρχουν περιπτώσεις ιδιοτήτων των δεδομένων που είναι πιθανό να μην μπορούν να ελεγχθούν με ακριβή τρόπο, ώστε να ταιριάζουν πιο κοντά στις ιδιότητες των πραγματικών δεδομένων καταγραφής. Ιδανικά για την ανάλυση και τη δημιουργία του μοντέλου μηχανικής μάθησης, θα έπρεπε να υπάρχει ένας μεγάλος αριθμός παρατηρήσεων από διαφορετικούς χρήστες ώστε να υπάρχει διαφοροποίηση στα δεδομένα σε επίπεδο συνηθειών, ώστε να μπορεί και το μοντέλο να αναγνωρίσει τις συσχετίσεις μεταξύ των δεδομένων πιο αποδοτικά.



5.3.1 Διαγράμματα κατανομών των δεδομένων που δημιουργήθηκαν

Στις παραπάνω εικόνες φαίνονται κάποιες κατανομές από τα δεδομένα που δημιουργήθηκαν βάσει των υποθέσεων, παραδοχών και περιορισμών που έχουν εφαρμοστεί. Αυτό που μπορεί να παρατηρηθεί στα δεδομένα είναι η έντονη εφαρμογή υποθέσεων και περιορισμών που στην πραγματικότητα, δηλαδή σε πραγματικά δεδομένα να έδιναν αρκετά διαφορετικές κατανομές.

Η δεύτερη υπόθεση έχει να κάνει με τα ίδια τα δεδομένα ως χαρακτηριστικά εισόδου. Μπορεί τα συγκεκριμένα δεδομένα να μην έχουν αρκετά δυνατή σύνδεση ώστε να μπορούν να παράγουν αποδοτικά το συμπέρασμα για τη δεδομένη έξοδο. Ακόμη μπορεί να μην είναι αρκετά ως χαρακτηριστικά και να χρειάζεται προσθήκη ή δημιουργία επιπλέον χαρακτηριστικών που να οδηγούν σε καλύτερο αποτέλεσμα. Εδώ πρέπει να τονισθεί ότι έχει γίνει η αποφυγή χρήσης, χαρακτηριστικών που μπορεί να επηρεάζουν σημαντικά την έξοδο (biased) οπότε να κυριαρχούν κατά την εκπαίδευση του μοντέλου έναντι των άλλων χαρακτηριστικών και επομένως οι προβλέψεις να εξαρτώνται άμεσα και μόνο από αυτά. Τέτοια χαρακτηριστικά για την συγκεκριμένη περίπτωση είναι η συνολική απόσταση, η συνολική διάρκεια ακόμα και η μέση ταχύτητα του χρήστη.

Σχετικά με τις άλλες μεθόδους που δοκιμάστηκαν, ήταν η πρόβλεψη κατηγορίας βημάτων ως πρόβλημα ταξινόμησης, κατανέμοντας τον αριθμό βημάτων σε τρεις διαφορετικές κατηγορίες με την έννοια της δραστηριότητας χαμηλή – μεσαία – υψηλή. Εδώ το μοντέλο, έδωσε ως αποτέλεσμα F1 score ~0.75. Σε άλλη περιπτώσεις που πραγματοποιήθηκαν κάποιοι μετασχηματισμοί στα χαρακτηριστικά του μοντέλου όπως η μείωση της ακρίβειας των latitude και longitude σε δύο δεκαδικά

ψηφία ή η δημιουργία clusters τοποθεσιών και χρήση τους ως τέτοια, έδωσαν παρόμοια αποτελέσματα με αυτά που αναφέρθηκαν και νωρίτερα.

Τελικά με βάση τα αποτελέσματα των δοκιμών, γνωρίζοντας ότι τα πραγματικά δεδομένα θα εμφάνιζαν διαφορετικές κατανομές αλλά και μοτίβα, από αυτά που δημιουργήθηκαν χειροκίνητα, βγαίνει το συμπέρασμα ότι θα μπορούσε να γίνει βελτιστοποίηση της αρχιτεκτονικής και ρύθμισης των παραμέτρων του μοντέλου αν παρέχονταν σε αυτό πραγματικά δεδομένα προς εκπαίδευση, μιας και θα υπήρχε ανεξαρτησία στις παρατηρήσεις αλλά και θα μπορούσαν ενδεχομένως να αποκλειστούν και χαρακτηριστικά που επηρεάζουν αρκετά την απόδοση του μοντέλου ή δημιουργηθούν νέα χαρακτηριστικά.

7. Μελλοντικές Επεκτάσεις

Η υπάρχουσα έρευνα και υλοποίηση ενός ολοκληρωμένου συστήματος, όπως αυτό περιεγράφηκε παρέχει αρκετό περιθώριο προς μελλοντικές επεκτάσεις. Επεκτάσεις που αφορούν την εφαρμογή android και που κατά συνέπεια επηρεάζουν και τα υπόλοιπα συστήματα, θα μπορούσε να είναι η προσθήκη εξατομικευμένων χαρακτηριστικών για τον χρήστη, όπως η περιοχή του ή οι περιοχές που του αρέσουν να επισκέπτεται για περπάτημα ή κάποια άλλη δραστηριότητα, χαρακτηριστικά όπως το βάρος του, η ηλικία του και οι συνήθειες φυσικής του δραστηριότητας ως κατηγορία φυσικής δραστηριότητας. Αυτά τα δεδομένα θα μπορούσαν να έχουν κάποια συσχέτιση με τον αριθμό βημάτων που επιτελεί σε κάθε διαδρομή του, συνεπώς θα μπορούσαν να συμβάλουν σε εξατομικευμένες προβλέψεις από το μοντέλο μηχανικής μάθησης.

Άλλη μια προσθήκη που θα μπορούσε στο μέλλον να ενσωματωθεί είναι η προβολή προτάσεων για τοποθεσίες σε ένα καινούριο Fragment, τοποθεσίες που επισκέπτονται άλλοι χρήστες με παρόμοιες συνήθειες και εξατομικευμένα χαρακτηριστικά.

Τελικώς, πρέπει να αναφερθεί ότι η συγκεκριμένη υλοποίηση παρέχει τη δυνατότητα αναβάθμισης και επανεκπαίδευσης του μοντέλου, όσο δημιουργούνται νέα δεδομένα ή νέοι συνδυασμοί δεδομένων. Με την αναβάθμιση του μοντέλου αυτό μπορεί εύκολα να ενσωματωθεί εκ νέου στην εφαρμογή Android με τη μορφή Application Update, κάθε φορά που αναβαθμίζεται το μοντέλο Tensorflow Lite.

8. Σύνοψη

Συνοψίζοντας το παρόν έγγραφο, πρέπει να αναφερθεί ότι η θεματική ενότητα με την οποία καταπιάνεται η έρευνα και η υλοποίηση της συγκεκριμένης εργασίας, ανέδειξε τη σημασία χρήσης σύγχρονων μεθόδων και τεχνικών, για τον συνδυασμό τεχνολογιών για την εξαγωγή συμπερασμάτων στους χρήστες με άμεσο τρόπο. Η Επιστήμη Δεδομένων καθημερινά, ολοένα και περισσότερο αξιοποιείται σε διάφορους τύπους εφαρμογών, ενισχύοντας την εμπειρία χρήσης και παρέχοντας χρήσιμες συμβουλές και προτάσεις προς τους χρήστες.

Συνεπώς στην παρούσα εργασία ερευνήθηκαν, αναλύθηκαν και υλοποιήθηκαν οι μέθοδοι και τεχνικές με τις οποίες μπορεί να συνδυαστεί μια εφαρμογή android με ένα μοντέλο μηχανικής μάθησης σε μια αμφίδρομη διαδικασία, όπου η εφαρμογή παράγει δεδομένα, αυτά τα δεδομένα αποτελούν το υλικό τροφοδοσίας ενός μοντέλου μηχανικής μάθησης και αυτό το μοντέλο με τη σειρά του επιστρέφει πληροφορία πίσω στην εφαρμογή android.

Επιπλέον πραγματοποιήθηκε η έρευνα και η εκμάθηση σε μια σειρά από καινούριες τεχνικές όσον αφορά την υλοποίηση εφαρμογών τόσο σχετικά με το οικοσύστημα του Android, το οποίο συνεχώς εξελίσσεται και φέρνει νέα εργαλεία και μεθόδους, σύγχρονα και πιο αποδοτικά, όσο και σχετικά με το .NET Framework το οποίο αποτελεί ένα από τα πιο δημοφιλή και αποδοτικά frameworks για υλοποίηση εφαρμογών ιστού, όσο και με τεχνικές και βιβλιοθήκες που χρησιμοποιούνται ευρέως στον χώρο της επιστήμης δεδομένων.

Βιβλιογραφία

1. Maria Virvou, Efthimios Alepis, George A. Tzihrintzis, Lakhmi C. Jain - Springer Cham (2020) **"Machine Learning Paradigms"**

2. Efthimios Alepis, Constantinos Patsakis (2019) **"Unravelling Security Issues of Runtime Permissions in Android"**

3. Constantinos Patsakis, Efthimios Alepis (2018) **"Knock-Knock: The Unbearable Lightness of Android Notifications"**

4. Achilleas Papageorgiou, Michael Strigkos, Eugenia Politou, Efthimios Alepis, Agusti Solanas, Constantinos Patsakis (2018) **"Security and Privacy Analysis of Mobile Health Applications: The Alarming State of Practice"**

5. Eugenia Politou, Efthimios Alepis, Constantinos Patsakis (2017) **"A survey on mobile affective computing"**

6. Efthimios Alepis, Maria Virvou, Polychronis Kontomaris (2021) **"Covid-19 Mobile Tracking Application Utilizing Smart Sensors"**

7. Efthimios Alepis, Maria Virvou, Savas Drakoulis (2014) **"Human smartphone interaction: Exploring smartphone senses"**

8. Spyridoula Vassilopoulou, Vassilis Sakkas, Joseph Batsos, Evangelos Sakkopoulos, Dimitrios Ringas (2020) **"Geo-Environmental Information Management Software for Automated Terrain and Ground Deformation Analysis"**

9. Aristea Kontogianni, Efthimios Alepis, Maria Virvou, Constantinos Patsakis – Springer Cham, (2024) **"Smart Tourism–The Impact of Artificial Intelligence and Blockchain"**

10. Emmanuel Stamatakis, Matthew Ahmadi, Marie H Murphy, Timothy James Chico, Karen Milton, Borja Del Pozo Cruz, Peter T Katzmarzyk, I-Min Lee, Jason Gill (2023) **"Journey of a thousand miles: from 'Manpo-Kei' to the first steps-based physical activity guidelines"**

11. I-Min Lee, Eric J. Shiroma, Masamitsu Kamada, et al (2019) **"Association of Step Volume and Intensity With All-Cause Mortality in Older Women"**

12. Salman Ahmed, Qamar Mahmood (2020) "**An authentication-based scheme for applications using JSON web token"**

13. László Viktor Jánoky, János Levendovszky, Péter Ekler (2018) **"An analysis on the revoking mechanisms for JSON Web Tokens"**

14. Anastasios Kountouris, Evangelos Sakkopoulos (2018) **"Survey on Intelligent Personalized Mobile Tour Guides and a Use Case Walking Tour App"**

15. **"Request Runtime Permissions Documentation"** Android Developers
https://developer.android.com/training/permissions/requesting

16. **"Kotlin Coroutines on Android Documentation"** Android Developers
https://developer.android.com/kotlin/coroutines

17. **"Android Preferences DataStore Documentation"** Android Developers
https://developer.android.com/topic/libraries/architecture/datastore

18. **"Android Services Documentation"** Android Developers
https://developer.android.com/develop/background-work/services

19. **"Retrofit Documentation" Square**
https://square.github.io/retrofit/

20. **"OkHttp Documentation" Square**
https://square.github.io/okhttp/

21. **"Adding a TFLite model to your android app"** Terrence Aluda
https://dev.to/agusioma/adding-a-tflite-model-to-your-android-app-3m5c

22. **"Open Weather Maps API Documentation"** OpenWeatherMaps
https://openweathermap.org/api/one-call-3

23. **"Microsoft .NET Core Web API Documentation"** Microsoft
https://learn.microsoft.com/en-us/aspnet/core/web-api/?view=aspnetcore-8.0

24. **"Microsoft Entity Framework Core"** Microsoft
https://learn.microsoft.com/en-us/ef/core/

25. **"Containerize a .Net App"** Microsoft
https://learn.microsoft.com/en-us/dotnet/core/docker/build-container

26. **"Docker Compose Documentation"** Dockerdocks – Docker
https://docs.docker.com/compose/

27. **"Tensorflow – Keras Documentation"** Tensorflow
https://www.tensorflow.org/guide/keras

28. **"Scikit-learn Data Preprocessing"** Scikit-learn Documentation
https://scikit-learn.org/stable/modules/preprocessing.html