ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Safe-R: Εφαρμογή εκτίμησης και αναπαράστασης της ποιότητας του οδοστρώματος

Πέτρος Β Μητάκος

Επιβλέπων: **Ευστάθιος Χατζηευθυμιάδης,** Αναπληρωτής Καθηγητής

ΑΘΗΝΑ

ΑΠΡΙΛΙΟΣ 2016

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Safer-R: Εφαρμογή εκτίμησης και αναπαράστασης της ποιότητας του οδοστρώματος

Πέτρος Β. Μητάκος
Α.Μ.: M1238

ΕΠΙΒΛΕΠΩΝ: **Ευστάθιος Χατζηευθυμιάδης,** Αναπληρωτής Καθηγητής

ΕΞΕΤΑΣΤΙΚΗ ΕΠΙΤΡΟΠΗ: **Όνομα Επώνυμο,** Τίτλος (π.χ Αναπληρωτής Καθηγητής)

Απρίλιος 2016

ΠΕΡΙΛΗΨΗ

Βασικός σκοπός της παρούσας διπλωματικής εργασίας αποτελεί η συλλογή, η εκτίμηση και η αναπαράσταση δεδομένων που αφορούν την ποιότητα του οδοστρώματος με χρήση κινητών συσκευών. Επίσης γίνεται μία παράθεση δεδομένων τροχαίων ατυχημάτων εντός της περιοχή δοκιμών ώστε να μελετηθεί πιθανώς συσχετισμός με την ποιότητα του οδοστρώματος. Έγινε υλοποίηση μίας Android εφαρμογής (Application) καθώς επίσης και ενός εξυπηρετητή (Server). Η εφαρμογή καλείται να συλλέξει δεδομένα ποιότητας του οδοστρώματος με χρήση του επιταχυνσιόμετρου καθώς και δεδομένα τοποθεσίας με χρήση του δέκτη GPS. Ο συνδυασμός θέσης και αναταραχής στέλνονται στον εξυπηρετητή όπου και γίνεται περαιτέρω επεξεργασία της πληροφορίας και εξαγωγή εκτιμήσεων για την ποιότητα του οδοστρώματος. Ο εξυπηρετητής αποστέλλει τα επεξεργασμένα δεδομένα ποιότητας πίσω στην εφαρμογή ώστε να γίνει αναπαράστασή τους, σε ζωντανό χρόνο, σε ηλεκτρονικό χάρτη τύπου OpenStreetMap. Τέλος, γίνεται μία απλή παράθεση δεδομένων τροχαίων ατυχημάτων επάνω από τα δεδομένα των δρόμων.

ΘΕΜΑΤΙΚΗ ΠΕΡΙΟΧΗ: Συλλογή & Επεξεργασία Χωρικών Δεδομένων

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: Ποιότητα Οδοστρώματος, Εφαρμογή Android, αναπαράσταση Χωρικά Δεδομένα, Τροχαία Ατυχήματα

ABSTRACT

Basic purpose of this thesis is the collection, assessment and representation of data related with road quality using mobile devices. Also, road accidents data are overlaid over the test area in order to research a possible correlation between road quality and accidents. An Android application was implemented as well as a Server. The application is collecting data regarding road quality, using the internal accelerometer, and positioning data, using the internal GPS. The combination of those two types of data is sent to Server, where they are further processed producing road quality assessments. Server sends processed data, regarding road quality, back to the application and those data are projected onto a map (OpenStreetMap). In the last step of the procedure road accidents data (points) are overlaid on top of processed data.

SUBJECT AREA: Spatial Data Collection & Processing

KEYWORDS: Road Quality, Android Application, Spatial Data, Road Accidents

ΠΕΡΙΕΧΟΜΕΝΑ

ΚΑΤΑΛΟΓΟΣ ΕΙΚΟΝΩΝ

ΠΡΟΛΟΓΟΣ

Η εργασία εκπονήθηκε στα πλαίσια της διπλωματικής εργασίας του 6ου κύκλου Μεταπτυχιακού Προγράμματος Σπουδών (Τεχνολογίες Πληροφορικής και Επικοινωνιών). Ο τόπος συλλογής όλων των δεδομένων είναι η ευρύτερη της Πανεπιστημιούπολης Αθηνών (περιοχές Ζωγράφου, Καισαριανή, Χολαργός, Αμπελόκηποι).

1. ΕΙΣΑΓΩΓΗ

1.1 Ορισμός αρχικού προβλήματος

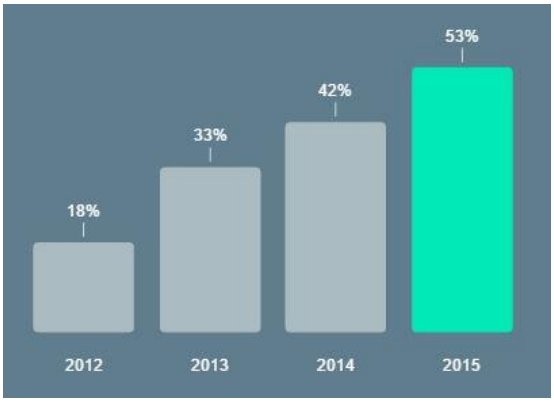
Σύμφωνα με τον Παγκόσμιο Οργανισμό Υγείας (ΠΟΥ) η Ελλάδα κατέχει την πρώτη θέση στα θανατηφόρα τροχαία ατυχήματα, ανάμεσα στις χώρες της Ευρωπαϊκής Ένωσης, με 9,1 θανάτους ανά 100.000 πολίτες (World Health Organization, 2015).

Ανάμεσα στις διάφορες αιτίες που μπορεί να προκαλέσουν ένα τροχαίο υπάρχει και αυτή της ποιότητας του οδικού δικτύου. Ειδικότερα όσων αφορά τα δίκυκλα, ένα κακό οδόστρωμα μπορεί να αποτελέσει αιτία απώλειας ελέγχου του οχήματος και τραυματισμού.

Έχει ήδη πραγματοποιηθεί μία προσπάθεια ανάλυσης του οδοστρώματος μέσω της μεθόδου της Υπερφασματικής Τηλεσκόπησης (Ανδρέου, 2008). Μέσω της συγκεκριμένης μεθόδου έγινε μια προσπάθεια χρονολόγησης του οδοστρώματος. Παρόλα αυτά η χρονολόγηση αποτελεί μόνο ένα παράγοντα ποιότητας. Η ομαλότητα του οδοστρώματος αποτελεί έναν ακόμη σημαντικό παράγοντα και είναι και ο βασικός παράγοντας που θα μελετηθεί στην παρούσα εργασία.

Η διαδικασία επισκευής των δρόμων δε θα έπρεπε να βασίζεται μόνο στην παλαιότητα τους αλλά και στην ουσιαστική κατάστασή τους έπειτα απο την καθημερινή τους χρήση και κατασκευή δημοσίων έργων κατά μήκους τους. Έχοντας, λοιπόν, έναν ηλεκτρονικό χάρτη καταγραφής της ομαλότητας του οδοστρώματος θα μπορούσε να δωθεί η δυνατότητα της έγκαιρης και στοχευμένης επιδιόρθωσής τους από τις αρμόδιες υπηρεσείες.

Η διάδοση και χρήση σύγχρονων "έξυπνων" κινητών συσκευών (Smartphone) έχει φτάσει σε αρκετά υψηλά επίπεδα ακόμη και στην Ελλάδα (Εικόνα 1). Η συντριπτική πλειοψηφία των Smartphone πλέον έχει δυνατότητες λήψης της θέσης (Δέκτης GPS - GPS Receiver) καθώς επίσης και υπολογισμού των επιταχύνσεων που δέχεται η συσκευή (Επιταχυνσιόμετρο - Accelerometer).



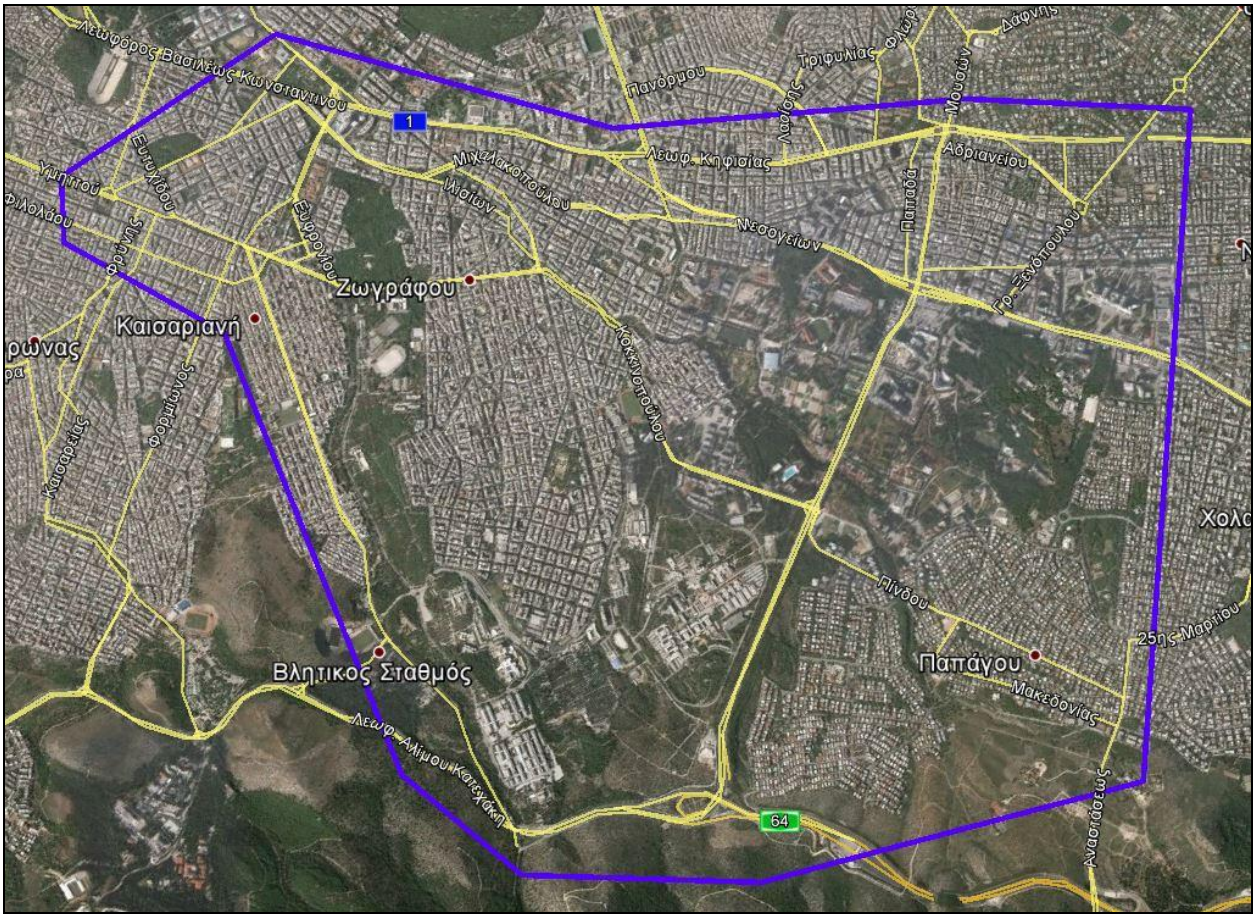
Εικόνα 1 - Χρήση Συσκευών Smartphone στην Ελλάδα

Υπάρχει η δυνατότητα να γίνει χρήση του επιταχυνσιόμετρου των κινητών συσκευών σε συνεργασία με το δέκτη GPS ώστε να γίνει συλλογή αναταραχών κατά την οδήγηση. Αυτά τα δεδομένα έπειτα από επεξεργασία τους μπορούν να αναπαρασταθούν σε χάρτη. Επομένως, κάθε χρήστης κινητού με αισθητήρες λήψης θέσης και επιτάχυνσης θα μπορούσε να αποτελέσει ένα κινητό αισθητήρα καταγραφής της ποιότητας του οδικού δικτύου.

Στα επόμενα κεφάλαια της παρούσας εργασίας γίνεται περιγραφή μίας τέτοιας προσπάθειας καταγραφής με τη δημιουργία της Android εφαρμογής με τίτλο Safe-R όπως επίσης και του εξυπηρετητή (server) με τίτλο Safe-House.

1.2 Περιοχή δοκιμών

Ως περιοχή δοκιμών ορίζεται η ευρύτερη περιοχή περιμετρικά της Πανεπιστημιούπολης του Ε.Κ.Π.Α.. Κατά τη διαδικασία της δειγματοληψίας επομένως καλύπτονται περιοχές που ανήκουν στους δήμους Ζωγράφου, Παπάγου, Χολαργού, Καισαριανής και Αθηναίων (Αμπελόκηποι). Ο χάρτης της περιοχής δοκιμών εμφανίζεται στην Εικόνα 2.



Εικόνα 2 - Περιοχή Δοκιμών

1.3 Βασικές παραδοχές

Καταρχάς θα πρέπει να ορισθεί κάτι πιο συγκεκριμένο ώστε να γίνει σημείο αναφοράς για την υπόλοιπη εργασία στο τι ακριβώς εννοούμε με τον όρο **ποιότητα οδοστρώματος**. Κάποιος μπορεί να θεωρήσει κακό ποιοτικά ένα δρόμο όταν αυτός έχει πολλές λακκούβες, ανωμαλίες, εάν δεν έχει αρκετή πρόσφυση λόγω παλαιότητας ή ακόμη και εάν έχει κακή σήμανση ή φωτισμό. Η παρούσα εργασία, όπως προαναφέρθηκε, αφορά καθαρά την καταγραφή της **ομαλότητας** του οδοστρώματος.

Κάθε ανωμαλία στο οδόστρωμα γίνεται αντιληπτή διαφορετικά ανάλογα με την ταχύτητα του οχήματος καταγραφής. Επίσης, μπορεί να μεταφράζεται με διαφορετικό τρόπο σε επιτάχυνση όταν η συσκευή καταγραφής βρίσκεται τοποθετημένη σε διαφορετικά σημεία του ίδιου οχήματος. Οπότε, γίνεται αντιληπτό πως οι παράμετροι που μπορούν να επηρεάσουν μία προσπάθεια αντικειμενικής καταγραφής είναι πάρα πολλές.

Στην παρούσα εργασία μελετούνται οι αναταραχές που δύναται να καταγράψει μία κινητή συσκευή τύπου smart phone και μέσω αυτής της καταγραφής να γίνει μία συσχέτιση με την ομαλότητα του οδοστρώματος. Όταν αναφερόμαστε σε smart phone δε θα πρέπει να παραλείπουμε το γεγονός ότι υπάρχει μία τεράστια ποικιλία συσκευών τύπου smart phone που έχουν εγκατεστημένο το λειτουργικό Android και έχουν δυνατότητες λήψης θέσης και επιταχύνσεων. Αυτό συνεπάγεται ποικιλία και σε βασικά όργανα λήψης των δεδομένων που μας απασχολούν. Η **ευαισθησία του γυροσκοπίου** όπως επίσης και **η ακρίβεια του δέκτη GPS** είναι δύο παράμετροι που δεν είναι εύκολο να μοντελοποιηθούν, όπως επίσης και σχεδόν απίθανο να έχουν την ίδια ευαισθησία από συσκευή σε συσκευή.

Για αυτό το λόγο, κατά τη διαδικασία των πειραμάτων, όλα τα δεδομένα καταγράφηκαν από την ίδια συσκευή Android, το Galaxy S4 Mini (I9195) της Samsung (Εικόνα 3). Με τη χρήση μίας συγκεκριμένης συσκευής βοηθούμε την αντικειμενικότητα και την ομοιογένεια των μετρήσεων.

Εικόνα 3 - Κινητή Συσκευή Καταγραφής

Βασικό στοιχείο στην ομοιογένεια της λήψης των δεδομένων αποτελούν το όχημα μέσα από το οποίο γίνεται η καταγραφή καθώς επίσης και η θέση της συσκευής καταγραφής κατά τη δειγματοληψία των αναταραχών του οδοστρώματος. Το όχημα παίζει το σημαντικότερο ρόλο καθώς διαφορετικά οχήματα έχουν διαφορετικούς τύπους αναρτήσεων "αντιλαμβάνονται" διαφορετικά μία ανωμαλία στο οδόστρωμα.

Επίσης, η συσκευή πρέπει να βρίσκεται σε συγκεκριμένη και σταθερή θέση κατά την καταγραφή. Αλλαγή στη θέση θα άλλαζε δραστικά τις μετρήσεις μας. Σε κάθε περίπτωση θα έπρεπε να υπάρξει ειδική μέριμνα έτσι ώστε να κατηγοριοποιηθούν τα διάφορα οχήματα και οι πιθανές θέσεις που μπορεί να βρίσκεται η συσκευή καταγραφής. Επειδή, όμως, κάτι τέτοιο αυξάνει εκθετικά το βαθμό ποικιλίας των δεδομένων, κατ' επέκταση και του αποθηκευτικού χώρου, επιλέχθηκε ένα συγκεκριμένο

όχημα και ένας συγκεκριμένος τρόπος τοποθέτησης της συσκευής για τη διαδικασία της πειραματικής καταγραφής της ποιότητας του δρόμου.

Το όχημα είναι ένα αυτοκίνητο της εταιρείας Opel και συγκεκριμένα το μοντέλο Astra (κατασκευής 2010 - Εικόνα 4) και η κινητή συσκευή τοποθετήθηκε σε σταθερή βάση πίσω από τη θέση του χειρόφρενου, ανάμεσα από τις θέσεις οδηγού και συνοδηγού. Έτσι έχουμε μετρήσεις επιταχύνσεων από ένα σημείο κοντά στο κέντρο βάρους του οχήματος και μπορούμε να θεωρήσουμε ότι οι καταγραφές είναι αντικειμενικές ως προς το πως μεταφράζεται η ποιότητα του οδοστρώματος σε επιταχύνσεις στη συσκευή.

Εικόνα 4 - Όχημα Καταγραφής

Τέλος, η ταχύτητα με την οποία γίνονται οι μετρήσεις είναι ικανή να δώσει διαφορετικά αποτελέσματα. Έγινε προσπάθεια ούτως ώστε η παράμετρος της ταχύτητας κατά τη διάρκεια λήψης των δειγμάτων να είναι ανάλογη των δρόμων και της τυπικής ταχύτητας που κινούνται τα οχήματα σε αυτούς. Κατά την επεξεργασία των δειγμάτων έγινε υπολογισμός της μέσης ταχύτητας που παρουσιάζουν τα δείγματα στον εκάστοτε δρόμο και αυτή η μέση ταχύτητα ορίζεται ως η ταχύτητα καταγραφής του δρόμου.

Δεδομένου ότι έχουμε τρείς βασικές παραμέτρους της καταγραφής της ποιότητας (**συσκευή**, **όχημα**, **θέση συσκευής**) σταθερές κατά τη διαδικασία των πειραμάτων αλλά και την **ταχύτητα** ορισμένη και ελεγχόμενη, μπορούμε να θεωρήσουμε ότι η καταγραφή όλων των δρόμων θα γίνει υπό παρόμοιες συνθήκες και επομένως θα προκύψει ένας αρτιότερος και πιο ακριβής τελικός χάρτης συμπερασμάτων.

Εξάλλου αυτό που περιμένουμε να προκύψει ως αποτέλεσμα τελικά δεν είναι ένας χάρτης που θα εμφανίζει το ακριβές μέγεθος μίας λακκούβας αλλά ***ένας χάρτης που θα επισημαίνει και θα κατηγοριοποιεί χρωματικά δρόμους εντός μίας περιοχής δοκιμών έχοντας ως κριτήριο το σύνολο των επιταχύνσεων τις οποίες καταγράφει η συσκευή Android καθώς το όχημα κινείται, σε μία ορισμένη ταχύτητα, κατά μήκος τους***.

2. Κινητή Συσκευή Καταγραφής Android

Στο παρόν κεφάλαιο θα δοθούν τα βασικά χαρακτηριστικά μίας τυπικής Android συσκευής smart phone. Αρχικά, θα περιγραφεί το ίδιο το λειτουργικό σύστημα και στη συνέχεια θα περιγραφούν αναλυτικά τα υποσυστήματα καταγραφής των επιταχύνσεων και της θέσης. Κυριότερος σκοπός του κεφαλαίου, επομένως, αποτελεί η περιγραφή των στοιχείων εκείνων της εφαρμογής που σχετίζονται άμεσα με το hardware της συσκευής και συνδράμουν στη διαδικασία καταγραφής του δρόμου.

2.1 Λειτουργικό Σύστημα

Το Android είναι λειτουργικό σύστημα για συσκευές κινητής τηλεφωνίας (smart phones) το οποίο τρέχει τον πυρήνα του λειτουργικού Linux. Αρχικά αναπτύχθηκε από την Google και αργότερα από την Handset Alliance|Open Handset Alliance. (Open Handset Alliance, 2007) Επιτρέπει στους κατασκευαστές λογισμικού να συνθέτουν κώδικα με την χρήση της γλώσσας προγραμματισμού Java, ελέγχοντας την συσκευή μέσω βιβλιοθηκών λογισμικού ανεπτυγμένων από την Google. (Shankland, 2007) Το Android είναι κατά κύριο λόγο σχεδιασμένο για συσκευές με οθόνη αφής, όπως τα έξυπνα κινητά τηλέφωνα. Το Android είναι το πιο ευρέως διαδεδομένο λογισμικό στον κόσμο.

Η πρώτη παρουσίαση της πλατφόρμας Android έγινε στις 5 Νοεμβρίου 2007, παράλληλα με την ανακοίνωση της ίδρυσης του οργανισμού Open Handset Alliance, μιας κοινοπραξίας 48 τηλεπικοινωνιακών εταιριών, εταιριών λογισμικού καθώς και κατασκευής ηλεκτρονικών συσκευών (hardware), οι οποίες είναι αφιερωμένες στην ανάπτυξη και εξέλιξη ανοιχτών προτύπων στις συσκευές κινητής τηλεφωνίας. (Open Handset Alliance, 2007) Η Google δημοσίευσε το μεγαλύτερο μέρος του κώδικα του Android υπό τους όρους της Apache License, μιας ελεύθερης άδειας λογισμικού.

Η συσκευή Samsung Galaxy S4 Mini, που επιλέχθηκε ώστε να εκτελεστούν τα πειράματα της παρούσας εργασίας, έχει εγκατεστημένη την έκδοση 5.1.1 (API Level 22) του λογισμικού Android.

2.2 Επιταχυνσιόμετρο

Το **επιταχυνσιόμετρο** είναι μία ηλεκτρομηχανική συσκευή που έχει την ικανότητα να μετρά δυνάμεις επιτάχυνσης οι οποίες εφαρμόζονται στη συσκευή. Αυτές οι δυνάμεις μπορεί να είναι στατικές, όπως είναι η επιτάχυνσης της βαρύτητας, ή δυναμικές όταν προκαλούνται - προέρχονται από αλλαγές στην ταχύτητα η στην διεύθυνση της κίνησης (επιταχύνσεις, επιβραδύνσεις, στροφές).

Στην περίπτωση των έξυπνων κινητών τηλεφώνων το επιταχυνσιόμετρο έχει πολύ μικρές διαστάσεις και αποτελεί ένα μεμονωμένο ολοκληρωμένο κύκλωμα το οποίο συνήθως τοποθετείται στο κέντρο της συσκευής κοντά στο κέντρο βάρους της. Αποτελείται από τρεις ανεξάρτητους πυκνωτές ένα για κάθε άξονα από τους x, y και z. Αυτοί οι πυκνωτές αποτελούνται από ένα σταθερό και ένα κινούμενο μέρος. Το κινούμενο μέρος συμπεριφέρεται σαν ελατήριο και επηρεάζεται από τις επιταχύνσεις που δέχεται η συσκευή. Αλλαγές στην επιτάχυνση προκαλούν αυξομείωση της απόστασης μεταξύ κινούμενου και ακίνητου μέρους στους πυκνωτές επομένως και αλλαγή στην τάση. Αυτές οι αλλαγές τάσεις μας δίνουν και την επιτάχυνση ανά άξονα.

Η επικοινωνία με το επιταχυνσιόμετρο της συσκευής παρέχεται μέσω μία διεπαφής προγραμματισμού (API) που παρέχεται από τη Google (android.hardware). Στη συνέχεια περιγράφεται ο τρόπος με τον οποίο χρησιμοποιείται η διεπαφή αυτή από τη εφαρμογή Safe-R.

2.2.1 Η κλάση SaferAccelerometer

Δημιουργήθηκε μία κλάση η οποία έχει τον πλήρη έλεγχο της διαδικασίας επικοινωνίας με το επιταχυνσιόμετρο. Η κλάση ονομάστηκε SaferAccelerometer. Ο κώδικας που ακολουθεί αφορά τη δημιουργία ενός αντικειμένου τύπου SaferAccelerometer.

```
public SaferAccelerometer(Context context) {
    SensorManager sensorManager = (SensorManager)
context.getSystemService(Context.SENSOR_SERVICE);
    sensorManager.registerListener(
            this, sensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER),
            SAMPLING_RATE_IN_MICROSECONDS);
}
```

Το όνομα της constructor μεθόδου ακολουθεί το όνομα της κλάσης και είναι **SaferAccelerometer**. Αρχικοποιεί ένα αντικείμενο **SensorManager** τύπου **Accelerometer**.

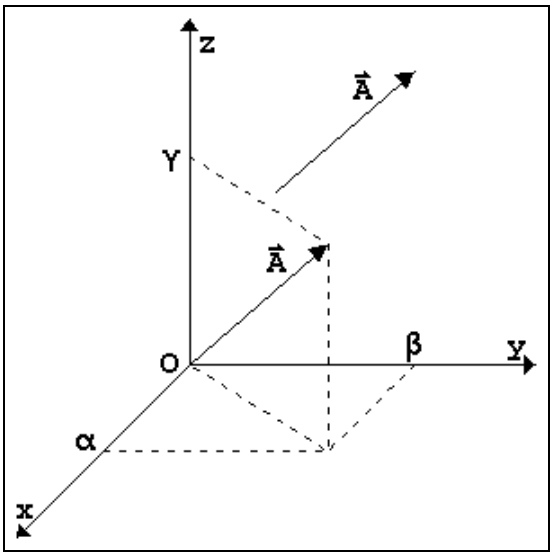
Η παράμετρος **SAMPLING_RATE_IN_MICROSECONDS** μας επιτρέπει να ορίσουμε τη συχνότητα δειγματοληψίας του αισθητήρα. Αυτό που ορίζουμε για την ακρίβεια είναι η καθυστέρηση που θέλουμε να υπάρχει μεταξύ δύο διαδοχικών μετρήσεων. Για την εξυπηρέτηση των αναγκών της εφαρμογής η καθυστέρηση έχει οριστεί στα 100.000 μsec. Αυτή η καθυστέρηση αντιστοιχεί σε **δειγματοληψία της τάξης των 10Hz**. Στο εγχειρίδιό του API γίνεται σαφές ότι αυτή η καθυστέρηση δεν είναι απόλυτα ακριβής οπότε αποτελεί την επιθυμητή καθυστέρηση της εφαρμογής. Τα αποτελέσματα μπορεί να είναι περισσότερες δειγματοληψίες ανά δευτερόλεπτο ή ακόμη και λιγότερες. Κατά τη δοκιμαστική λειτουργία της εφαρμογής παρατηρήθηκε απόκλιση από τα 10Hz κατά 4Hz. Οπότε οι μετρήσεις ανά δευτερόλεπτο είναι **13 - 14 Hz**. Το μέγεθος της απόκλισης υπάρχει πιθανότητα να αλλάζει από συσκευή σε συσκευή καθώς είναι άμεσα συνδεδεμένο με το είδος του επιταχυνσιόμετρου που συμπεριλαμβάνεται στην εκάστοτε συσκευή.

Στη βιβλιογραφία αναφέρεται πως όσο μικρότερη είναι αυτή η καθυστέρηση τόσο περισσότερο θα αυξάνεται ο φόρτος της συσκευής στην προσπάθεια συλλογής δεδομένων επιτάχυνσης. Όποτε έχοντας ορίσει εκ των προτέρων ένα ρυθμό συλλογής

και μάλιστα στο προαναφερθέν μέγεθος δεν υπερφορτώνουμε τον επεξεργαστή της συσκευής.

2.2.2 Υπολογισμός μέτρου της επιτάχυνσης

Τα τυπικά δεδομένα που μας επιστρέφει η διεπαφή του επιταχυνσιόμετρου περιγράφονται από τρεις παραμέτρους όπου στην υλοποίηση μας ονομάζονται **x**, **y** και **z**. Κάθε μεταβλητή αντιπροσωπεύει και έναν άξονα στον οποίο εφαρμόζεται κάποια επιτάχυνση. Εφόσον μιλάμε για τρεις διαστάσεις έτσι προκύπτουν και τρεις μεταβλητές. Οι τιμές που λαμβάνουν οι τρεις αυτές μεταβλητές μπορούν να έχουν θετικό και αρνητικό πρόσημο κάνοντας έτσι δυνατή την περιγραφή οποιασδήποτε στατικής ή δυναμικής επιτάχυνσης στο χώρο. Έτσι, στην Εικόνα 5 μπορούμε να θεωρήσουμε μία τυχαία στιγμιαία αναπαράσταση των τιμών x, y και z όπου στην συγκεκριμένη εικόνα απεικονίζονται με α, β και γ αντίστοιχα. Το διάνυσμα επιτάχυνσης που προκύπτει αναπαριστάται με $\vec{A}$.



Εικόνα 5 - Διάνυσμα Επιτάχυνσης

Καθώς αναφερόμαστε σε μία προσπάθεια εκτίμησης της ποιότητας του οδοστρώματος αναμένουμε πολύ μεγάλη ποικιλία σε μέτρα και διευθύνσεις των διανυσμάτων της επιτάχυνσης. Η συλλογή και των τριών μεταβλητών (x, y και z) χωριστά θα επιβάρυνε τις διαδικασίες υπολογισμού και αποθήκευσης των δειγμάτων και αν και αποτελεί μία λεπτομερή αναπαράσταση των επιταχύνσεων που δέχεται η συσκευή δεν επιλέχθηκε. Κρίθηκε ιδανικότερη λύση να γίνεται υπολογισμός του μέτρου της επιτάχυνσης και αυτό να αποτελεί το αποτέλεσμα της δειγματοληψίας. Αυτά τα μέτρα της επιτάχυνσης μπορούν να περιγράψουν ανωμαλίες στο οδόστρωμα καθώς ένα κινούμενο όχημα σε ανώμαλο δρόμο παρουσιάζει πολλές αλλαγές στις επιταχύνσεις που δέχεται και κατ' επέκταση και στα μέτρα των επιταχύνσεων που δέχεται ανά δευτερόλεπτο. Σε μία προσπάθεια ορθότερης και πληρέστερης "ανάγνωσης" του οδοστρώματος συνυπολογίζονται η **μέγιστη τιμή** των επιταχύνσεων, η **μέση τιμή** και η **διασπορά** τους ανά δευτερόλεπτο.

Επομένως, έχουμε την ανάγκη υπολογισμού αρχικά του μέτρου της επιτάχυνσης και στη συνέχεια των τριών παραμέτρων που αναφέρθηκαν προηγουμένως κάθε φορά που γίνεται δειγματοληψία (μέγιστη - μέση τιμή και διασπορά).

Ο υπολογισμός του μέτρου, σύμφωνα με τη θεωρία των διανυσμάτων και με χρήση του Πυθαγορείου Θεωρήματος, προκύπτει με τον τύπο $|\vec{A}| = \sqrt{\alpha^2 + \beta^2 + \gamma^2}$, όπου τα α, β και γ για την υλοποίησή μας ονομάζονται x, y και z. Η μέθοδος που καλείται να υπολογίσει αυτό το μέτρο ονομάζεται **getMagnitude** και περιέχει τον παρακάτω κώδικα:

```
private double getMagnitude() {
    return Math.sqrt(
            this.x * this.x
            + this.y * this.y
            + this.z * this.z);
}
```

Δεδομένης της φύσης της δειγματοληψίας και έτσι ώστε να γίνει πιο αποδοτικός και γρήγορος ο υπολογισμός της μέσης τιμής και της διασποράς έγινε χρήση συναρτήσεων σταδιακού υπολογισμού τους. Έτσι με κάθε νέα μέτρηση επαναϋπολογίζονται ο μέσος όρος και η διασπορά χωρίς να υπάρχει η ανάγκη αποθήκευσης όλων των μετρήσεων κατά τη διαδικασία συλλογής τους. Παρακάτω δίνονται ο μαθηματικός τύπος και η μεταφορά τους σε κώδικα μέσα στην κλάση **SaferAccelerometer**.

Υπολογισμός Μέσου Όρου

$$\bar{x}_n = \frac{(n-1)\bar{x}_{n-1} + x_n}{n}$$

```
private void calculateAverage() {
    this.average = ((this.samplesCount - 1) * this.average + this.lastSample) /
this.samplesCount;
}
```

Υπολογισμός Διασποράς

$$s_n^2 = \frac{(n-2)}{(n-1)} s_{n-1}^2 + \frac{(x_n - \bar{x}_{n-1})^2}{n}, n > 1$$

```
private void calculateVariance() {
    this.samplesCount++;
    if (this.samplesCount > 1) {
        this.variance = ((this.samplesCount - 2) * this.variance / (this.samplesCount - 1))
                + Math.pow((this.lastSample - this.average), 2) / this.samplesCount;
    }
}
```

Στις δύο προηγούμενες μεθόδους το n εκφράζεται από τη μεταβλητή **samplesCount**, ο μέσος όρος $\bar{x}$ από τη μεταβλητή **average,** η τελευταία μέτρηση x_n από τη μεταβλητή **lastSample** και η διασπορά s_n^2 από τη μεταβλητή **variance**.

Η βασικότερη μέθοδος της κλάσης **SaferAccelerometer**, η οποία μας παρέχει και τις τιμές των x, y, z, είναι η **onSensorChanged** η οποία, στην υλοποίησή μας, καλείται κάθε 100.000 μsec ή αλλιώς κάθε 0,1 sec.

Η ίδια μέθοδος καλεί τις μεθόδους getMagnitude ώστε να υπολογίσει το μέτρο της επιτάχυνσης και στη συνέχεια καλεί τις **calculateVariance, calculateAverage** και **calculateMax** ώστε να υπολογιστούν η διασπορά, ο μέσος όρος και η μέγιστη τιμή των δειγμάτων της επιτάχυνσης.

```
public void onSensorChanged(SensorEvent event) {
    if (this.sensorIsStarted) {
        this.x = event.values[0];
        this.y = event.values[1];
        this.z = event.values[2];

        if (this.samplingSessionStarted) {
            this.lastSample = this.getMagnitude();
            this.samples.add(this.lastSample);
            calculateVariance();
            calculateAverage();
            calculateMax();
        }
    }
}
```

Ολόκληρος ο κώδικας της κλάσης SaferAccelerometer βρίσκεται στο ΠΑΡΑΡΤΗΜΑ.

2.2.3 Παρατηρήσεις

Έχοντας κατά νου ότι η επιτάχυνση της βαρύτητας εφαρμόζεται μόνιμα στη συσκευή θα υπάρχει πάντα, σε κατάσταση ηρεμίας, τιμή μέτρου επιτάχυνσης ίση με 9,81 - όση δηλαδή και η επιτάχυνση της βαρύτητας.

Αυτό σημαίνει ότι αναμένουμε τιμές ίσες με 9,81, μεγαλύτερες αλλά ακόμη και μικρότερες. Ίσες τιμές με τη βαρύτητα κυρίως σε περιπτώσεις ηρεμίας, μικρότερες σε περιπτώσεις που υπάρχουν επιταχύνσεις αντίθετες προς αυτή της βαρύτητας και τέλος μεγαλύτερες όταν υπάρχουν επιταχύνσεις προς τη φορά της βαρύτητας.

Στην παρούσα εργασία ενδιαφερόμαστε κυρίως για τις τιμές που υπερβαίνουν αυτή της βαρύτητας και εκφράζουν έντονη αναταραχή πιθανώς λόγω ανωμαλίας στο οδόστρωμα και έντονης καθοδικής κίνησης του οχήματος προς τη φορά της επιτάχυνσης της βαρύτητας.

Ο λόγος για τον οποίο συλλέγουμε εκτός της μέτρησης του μέτρου της επιτάχυνσης και της μέσης τιμής, της διασποράς και της μέγιστης τιμής ανά δευτερόλεπτο είναι ακριβώς το γεγονός ότι επιλέχθηκε η χρήση του μέτρου της επιτάχυνσης και όχι του διανύσματος. Συνεκτιμώντας όλες τις παραμέτρους γίνεται δυνατή μία ορθότερη κατηγοριοποίηση των δρόμων με βάση την ομαλότητά τους.

2.3 Δέκτης GPS

Οι δέκτες GPS έχουν κατασκευαστεί εδώ και δεκαετίες για χρησιμοποιούνται παγκοσμίως έτσι ώστε να βοηθούν στον εντοπισμό της θέσης ακίνητου ή κινούμενου χρήστη. Βασικός και κοινός τρόπος λειτουργίας όλων των δεκτών είναι ο υπολογισμός της θέσης τους με μέτρηση απόστασής τους με τουλάχιστον τριών από τους 24 δορυφόρους που αποτελούν τον "πλέγμα" δορυφόρων στους οποίους στηρίζεται η τεχνολογία GPS.

Η τοποθέτηση δεκτών GPS και στις κινητές έξυπνες συσκευές (smart phones) θεωρείται πλέον δεδομένη. Σχεδόν όλες οι νέες κινητές συσκευές έχουν ενσωματωμένο ένα αντίστοιχο δέκτη έχοντας έτσι τη δυνατότητα εντοπισμού της θέσης της συσκευής με απόκλιση μερικών μέτρων.

Ακριβώς όπως και στην περίπτωση του επιταχυνσιόμετρου έτσι και στην περίπτωση του δέκτη GPS υπάρχει η κατάλληλη βιβλιοθήκη για επικοινωνία και ελέγχου του. Στην επόμενη ενότητα περιγράφεται η κλάση της εφαρμογής Safe-R η οποία δημιουργήθηκε για αυτό το λόγο.

2.3.1 Η κλάση SaferLocationManager

Για να γίνει πιο εύκολη η επικοινωνία με το δέκτη GPS της συσκευής δημιουργήθηκε η κλάση **SaferLocationManager**. Η κλάση αυτή επεκτείνει την κλάση του συστήματος Android **Service** έτσι ώστε να δίνεται η δυνατότητα στην κλάση να τρέχει ως υπηρεσία όσο η εφαρμογή είναι σε λειτουργία. Η κλάση παρέχει δυνατότητες εντοπισμού της θέσης της συσκευής, της ακρίβειας του δέκτη καθώς επίσης και της ταχύτητας.

Ο constructor της κλάσης είναι ο ακόλουθος:

```
protected LocationManager locationManager;

public SaferLocationManager(Context context) {
    try {
        locationManager = (LocationManager) context.getSystemService(LOCATION SERVICE);
        locationManager.requestLocationUpdates(
                LocationManager.GPS_PROVIDER,
                SAMPLING RATE IN MILLISECONDS,
                MIN DISTANCE CHANGE FOR UPDATES IN METERS, this);
    } catch (Exception e) {
        Log.e("Error : Location", "Impossible to connect to LocationManager", e);
    }
}
```

Αρχικοποιείται μια μεταβλητή τύπου LocationManager. Η κλάση LocationManager είναι εφοδιασμένη με αρκετές βοηθητικές μεθόδους επικοινωνίας με το δέκτη GPS της συσκευής. Μετά την αρχικοποίηση της απλώς γίνεται ρύθμιση των βασικών της παραμέτρων οι οποίες αφορούν τον τρόπο με τον οποίο επιθυμούμε να λαμβάνονται τα δεδομένα θέσης (GPS_PROVIDER), το ρυθμό με τον οποίο θέλουμε να γίνεται ανανέωση των δεδομένων θέσης στο δέκτη και η ελάχιστη μετακίνηση της συσκευής έτσι ώστε να γίνει αυτόματη ανανέωση της θέσης.

Οι δύο τελευταίες παράμετροι στην ουσία αλληλεπικαλύπτονται καθώς είτε θα παρέλθει το ελάχιστο διάστημα ώστε να γίνει ανανέωση δεδομένων θέσης είτε θα έχει υπάρξει μετατόπιση της συσκευής πάνω από το όριο που έχουμε δώσει. Στην περίπτωση της εφαρμογής μας δόθηκαν ως ρυθμός εντοπισμού της θέσης τα 500ms και ως ελάχιστη μετατόπιση τα 2 μέτρα. Έτσι, είμαστε σε θέση να θεωρήσουμε πως ανά δευτερόλεπτο θα έχουμε μία τουλάχιστον εκτίμηση της θέσης της συσκευής.

Ο τρόπος με τον οποίο γίνεται ενημέρωση και της εφαρμογής για τη θέση της συσκευής είναι αρκετά απλός και ως βασικότερη μέθοδος της κλάσης μπορεί να θεωρηθεί η μέθοδος **updateLocation** η οποία απλώς ελέγχει εάν ο δέκτης GPS είναι

ενεργοποιημένος και εν συνεχεία ζητάει από τον LocationManager τα νέα δεδομένα θέσης καλώντας την **getLastKnownLocation**. Η τελευταία επιστρέφει ένα αντικείμενο τύπου **Location**, το οποίο περιέχει πληροφορίες όπως γεωγραφικό μήκος/πλάτος, ακρίβεια GPS και ταχύτητα.

```
public void updateLocation() {
    if (canGetLocation()) {
        location = locationManager.getLastKnownLocation(LocationManager.GPS_PROVIDER);
        updateLocationData();
    }
}

private void updateLocationData() {
    if (location != null) {
        latitude = location.getLatitude();
        longitude = location.getLongitude();
        accuracy = location.getAccuracy();
        speed = location.getSpeed();
    }
}
```

Στη συνέχεια καλείται η μέθοδος **updateLocationData**. Οι τιμές του αντικειμένου location ανατίθενται σε τοπικές μεταβλητές της κλάσης μας για περαιτέρω επεξεργασία στη συνέχεια.

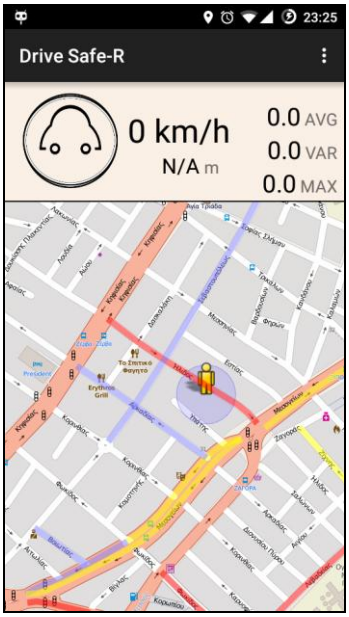
Ολόκληρος ο κώδικας της κλάσης SaferLocationManager βρίσκεται στο ΠΑΡΑΡΤΗΜΑ.

3. Η Εφαρμογή Συλλογής Δεδομένων, Safe-R

Στο παρόν κεφάλαιο γίνεται παρουσίαση του σχεδιασμού, της λειτουργίας και μερικών βασικών λεπτομερειών που αφορούν την κατασκευή της Android εφαρμογής συλλογής δεδομένων - **Safer-R**. Δίνονται οι βασικές μορφές δομών δεδομένων που καλείται να δημιουργήσει, να αποθηκεύσει και να αποστείλει η εφαρμογή (**Safer-R**) στον server (Safe-House). Τέλος, παρουσιάζονται κάποια προβλήματα που προέκυψαν κατά τη κατασκευή και τη δοκιμή της εφαρμογής καθώς και ο τρόπος που αυτά αντιμετωπίστηκαν.

3.1 Εμφάνιση

Η κεντρική ιδέα της εφαρμογής είναι η αναπαράσταση της ποιότητας του οδοστρώματος σε χάρτη. Οπότε η αρχική και κεντρική σελίδα της περιέχει ένα χάρτη στο μεγαλύτερο μέρος της όπως φαίνεται και από την Εικόνα 6. Στο χάρτη μπορεί ο χρήστης να δει με τρία διαφορετικά χρώματα (μπλε, κίτρινο, κόκκινο) τις διάφορες καταστάσεις του οδοστρώματος. Με μπλε απεικονίζονται οι δρόμοι που θεωρούνται ομαλοί, με κίτρινο οι δρόμοι με λίγες αναταραχές και με κόκκινο οι δρόμοι που έχουν χαρακτηριστεί ως αρκετά ανώμαλοι από την εφαρμογή. Ανάλυση του τρόπου με τον οποίο γίνεται η χρωματική αυτή αναπαράσταση γίνεται στην ενότητα 3.6.4.



Εικόνα 6 - Αρχική Σελίδα Εφαρμογής Safe-R

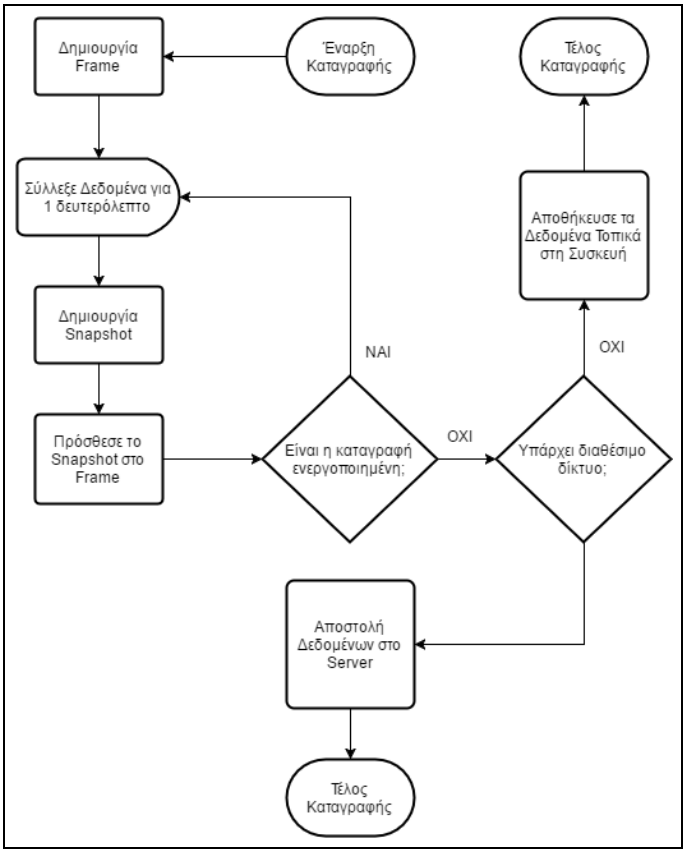
Ακριβώς πάνω από τον χάρτη υπάρχει ένα κυκλικό κουμπί, με το οποίο ξεκινάει η καταγραφή, και δεξιά υπάρχει η ένδειξη της ταχύτητας καθώς επίσης και η απόκλιση σε μέτρα για το δέκτη GPS. Επίσης δεξιά βρίσκονται και οι τρεις ενδείξεις από τα δείγματα επιτάχυνσης που καταγράφονται όταν η εφαρμογή βρίσκεται σε κατάσταση καταγραφής. Η ταχύτητα, η απόκλιση του GPS και οι ενδείξεις της επιτάχυνσης ενεργοποιούνται μόνο όταν είναι ενεργή η καταγραφή ενώ μηδενίζονται σε κατάσταση αδράνειας όπως στην προηγούμενη εικόνα.

Τέλος, επάνω δεξιά εμφανίζεται ένα σύμβολο το οποίο οδηγεί το χρήστη στη σελίδα των ρυθμίσεων της εφαρμογής. Μοναδική ρύθμιση της εφαρμογής είναι η δυνατότητα αλλαγής της διεύθυνσης του server (Safe House). Έτσι δεν απαιτείται νέα έκδοση της εφαρμογής κάθε φορά που υπάρχει αλλαγή στη διεύθυνση που αποστέλλονται τα δεδομένα.

3.2 Βασική αρχιτεκτονική

Η βασική λειτουργία της εφαρμογής είναι η συλλογή δεδομένων. Αυτή η δειγματοληψία, όταν ενεργοποιηθεί από τον χρήστη, πρέπει να βρίσκεται σε διαρκή λειτουργία και να συλλέγει δεδομένα θέσης και επιτάχυνσης. Οπότε, όταν ο χρήστης πατήσει το κουμπί πάνω αριστερά στην αρχική οθόνη ξεκινάει μία ατέρμονη επανάληψη κατά την οποία συλλέγονται δεδομένα για ένα δευτερόλεπτο, έπειτα αυτά αποθηκεύονται σε ένα μοντέλο που ονομάστηκε στιγμιότυπο (Snapshot) και όταν έρχεται η στιγμή ώστε να ξεκινήσει το επόμενο δευτερόλεπτο μετρήσεων αυτό το Snapshot αποθηκεύεται σε μία λίστα που ονομάστηκε πλαίσιο (Frame).

Όταν λήξει η περίοδος δειγματοληψίας γίνεται προσπάθεια αποστολής του πλαισίου (Frame) στο server (Safe House). Εάν υπάρχει δίκτυο διαθέσιμο τότε αποστέλλονται τα δείγματα αλλιώς αποθηκεύονται τοπικά ώστε να επιχειρηθεί ξανά αποστολή τους στο μέλλον. Όλη η διαδικασία που περιγράφηκε εμφανίζεται και στην Εικόνα 7 παρακάτω.



Εικόνα 7 - Λειτουργία Καταγραφής

Παράλληλα με τη διαδικασία της καταγραφής υπάρχει και η διαδικασία της ανανέωσης του χάρτη των δεδομένων. Η λογική της ανανέωσης του χάρτη έχει ως εξής. Υπολογίζεται το πλαίσιο το οποίο είναι εμφανές και αποστέλλονται τα στοιχεία του πολυγώνου αυτού στο server. Ο server απαντάει με όσους δρόμους εμπεριέχονται ή τέμνονται από το συγκεκριμένο πολύγωνο μαζί με τα στοιχεία που είναι καταγεγραμμένα για τους συγκεκριμένους δρόμους. Οι δρόμοι και τα δεδομένα αυτά σχεδιάζονται δυναμικά πάνω στο χάρτη και εμφανίζονται με το αντίστοιχο χρώμα σε αυτόν.

Οι δύο προηγούμενες διεργασίες (συλλογή, εμφάνιση δεδομένων) αποτελούν δύο απόλυτα ανεξάρτητες διεργασίες. Υπάρχει η δυνατότητα ανανέωσης των δεδομένων του χάρτη είτε η εφαρμογή βρίσκεται σε κατάσταση καταγραφής είτε όχι.

3.3 Κυριότερες δομές δεδομένων & κλάσεις

Η λογική καταγραφής δεδομένων στην εφαρμογή έχει ως εξής. Κάθε δευτερόλεπτο έχουμε 14 καταγραφές του μέτρου της επιτάχυνσης (14 Hz) και δύο καταγραφές της θέσης της συσκευής (2 Hz). Όσο η συσκευή βρίσκεται σε κατάσταση καταγραφής συλλέγονται αυτά τα δεδομένα και στο τέλος της καταγραφής γίνεται προσπάθεια αποστολή τους στο server.

Επομένως κρίθηκε αναγκαία η κατασκευή δύο βασικών δομών δεδομένων για την αποθήκευση όλων των πληροφοριών με τρόπο πρακτικό και βολικό για τη μετέπειτα επεξεργασία τους είτε από την εφαρμογή είτε, αργότερα, στην Βάση Δεδομένων από τον server. Οι δύο δομές δεδομένων ονομάστηκαν **Snapshot** και **Frame** με αντίστοιχες κλάσεις τις **SaferSnapshot** και **SaferFrame**. Ακολουθεί περαιτέρω ανάλυση και επεξήγηση αυτών των δύο βασικών κλάσεων.

3.3.1 Η κλάση SaferSnapshot

Ένα αντικείμενο τύπου Snapshot αποτελεί τη συλλογή δεδομένων ενός μόνο δευτερολέπτου. Οι μεταβλητές της κλάσης **SaferSnapshot** βρίσκονται παρακάτω και αντιστοιχίζονται στις εξής πληροφορίες:

- **maxForce** - Η μέγιστη τιμή των δειγμάτων επιτάχυνσης.
- **avgForce** - Η μέση τιμή των δειγμάτων επιτάχυνσης.
- **varForce** - Η διακύμανση των δειγμάτων επιτάχυνσης.
- **speed** - Η ταχύτητα της συσκευής.
- **latitude** - Το γεωγραφικό πλάτος της συσκευής.
- **longitude** - Το γεωγραφικό μήκος της συσκευής.
- **accuracy** - Η ακρίβεια του GPS.

```
public class SaferSnapshot {
     private double maxForce;
     private double avgForce;
     private double varForce;
     private double speed;
     private double latitude;
     private double longitude;
     private double accuracy;

public SaferSnapshot(SaferAccelerometer accelerometer, SaferLocationManager locationManager) {
    this.avgForce = accelerometer.getAverage();
    this.maxForce = accelerometer.getMax();
    this.varForce = accelerometer.getVariance();
    this.speed = locationManager.getSpeed();
    this.latitude = locationManager.getLatitude();
    this.longitude = locationManager.getLongitude();
    this.accuracy = locationManager.getAccuracy();
}
```

Στον κατασκευαστή της κλάσης δίνονται αντικείμενα από τις δύο κλάσεις που περιγράφηκαν σε προηγούμενο κεφάλαιο - **SaferAccelerometer** και **SaferLocationManager**.

Στην κλάση SaferSnapshot υπάρχει και μία βασική μέθοδος η οποία διακρίνει κάθε Snapshot ξεχωριστά και ενημερώνει την εφαρμογή για το εάν θα πρέπει οι πληροφορίες του συγκεκριμένου Snapshot να αποθηκευθούν στην εφαρμογή και να αποσταλούν στο server. Αυτή η μέθοδος ονομάζεται **shouldBeAdded** και ο κώδικας της εμφανίζεται παρακάτω.

```
public boolean shouldBeAdded(List<LatLng> testAreaPolygon) {
    return (latitude != 0.0 && longitude != 0.0
            && accuracy > 0 && accuracy < SaferLocationManager.ACCURACY_THRESHOLD_IN_METERS
            && speed > SaferLocationManager.MIN_SPEED_THRESHOLD_IN_KM_PER_HOUR
            && speed < SaferLocationManager.MAX_SPEED_THRESHOLD_IN_KM_PER_HOUR
            && PolyUtil.containsLocation(new LatLng(latitude, longitude),testAreaPolygon, false)
    );
}
```

Οι παράμετροι που ελέγχονται είναι οι εξής:

- Υπάρχει έγκυρο γεωγραφικό μήκος και πλάτος.
- Η ακρίβεια του δέκτη είναι μεταξύ ενός και 10 μέτρων.
- Η ταχύτητα της συσκευής είναι μεταξύ 10 και 100 km/h.
- Η θέση της συσκευής είναι μέσα στα όρια της περιοχής δοκιμών.

Οι προηγούμενοι παράμετροι επιλέχθηκαν για πολύ συγκεκριμένους λόγους και για να μη γίνει κατάχρηση της Βάσης Δεδομένων από πληθώρα άχρηστων πληροφοριών.

Έγκυρο θεωρείται γεωγραφικό μήκος και πλάτος που δεν είναι ίσο με 0 και αυτό διότι με 0 αρχικοποιούνται οι τιμές του δέκτη όταν δεν έχει ακόμη υπολογίσει τη θέση της συσκευής.

Η ακρίβεια του GPS σε κατάσταση κανονικής λειτουργίας είναι τρία με πέντε μέτρα. Στην περίπτωση της εφαρμογής καταγραφής της ποιότητας του δρόμου δόθηκε ένα όριο έως τα δέκα μέτρα καθώς θεωρήθηκε μία λογική απόσταση λάθους για δεδομένα που αφορούν δρόμους. Ένας τυπικός δρόμος έχει πλάτος έξι με επτά μέτρα και δεδομένου ότι το όχημα καταγραφής της ποιότητας βρίσκεται στο κέντρο του δρόμου η απόκλιση έως 10 μέτρα θα κρατήσει τη μέτρηση αρκετά κοντά στο δρόμο υπό μελέτη ώστε να συμπεριληφθεί αργότερα η μέτρηση σε αυτόν. Παρόλα αυτά ακόμα και εάν ένα δείγμα είναι σε απόσταση μεγαλύτερη του ενός μέτρου από τον πλησιέστερο σε αυτό δρόμο, αυτό δεν λογίζεται στον τελικό υπολογισμό της ποιότητας.

Τέλος, όλα τα δεδομένα πρέπει να βρίσκονται εντός της περιοχή δοκιμών και ο τρόπος με τον οποίο υπολογίζεται κάτι τέτοιο είναι με χρήση της μεθόδου **containsLocation** και περνώντας ως παραμέτρους τα στοιχεία της θέσης (γεωγραφικό μήκος - πλάτος) και ένα πολύγωνο που ορίζει την περιοχή των δοκιμών.

Ο κώδικας της κλάσης SaferSnapshot βρίσκεται στο ΠΑΡΑΡΤΗΜΑ.

3.3.2 Η κλάση SaferFrame

Η κλάση SaferFrame μας δίνει τη δυνατότητα ομαδοποίησης των επιμέρους Snapshots. Έτσι κάθε φορά που ξεκινάει η διαδικασία της καταγραφής του οδοστρώματος γίνεται δημιουργία ενός νέου Frame το οποίο "γεμίζει" σταδιακά με Snapshot κάθε δευτερόλεπτο. Αυτό που προκύπτει είναι ένα Frame γεμάτο με τα Snapshots της τελευταίας καταμέτρησης. Ένα Frame είναι και αυτό που στέλνεται στο server για καταχώρηση και επεξεργασία.

Η κλάση έχει τρεις παραμέτρους. Μία λίστα με SaferSnapshots με το όνομα **snapshots** και μία μεταβλητή τύπου SaferFrameConfigs, η οποία περιέχει κάποια βασικά χαρακτηριστικά που περιγράφουν το εκάστοτε Frame όπως τον τύπο της συσκευής λήψης, τον τύπο του οχήματος και το id του χρήστη της εφαρμογής. Και, τέλος, το πλήθος των Snapshot που εμπεριέχονται στο συγκεκριμένο Frame.

```
private ArrayList<SaferSnapshot> snapshots;
private SaferFrameConfigs frameConfig;
private int validSnapshotsAmount = 0;

public SaferFrame() {
    snapshots = new ArrayList<>();
    config = new SaferFrameConfig(1, 1, 1, 0);
}
```

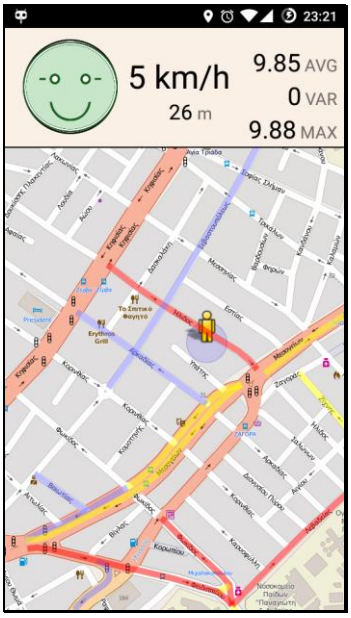
Οι παράμετροι της μεταβλητής **frameConfigs** υπάρχουν για μία πιθανή μελλοντική επέκταση της εφαρμογής Safe-R σε πολλούς χρήστες, οχήματα και συσκευές. Το πλήθος των Snapshots (**validSnapshotsAmount)**, παρόλα αυτά, αποστέλλεται ως ένας αριθμός επαλήθευσης από τον server έτσι ώστε να βεβαιωθεί η αποστολή όλων των δεδομένων και το γεγονός ότι όλα τα δεδομένα έχουν τη σωστή δομή ώστε να εισαχθούν στη Βάση Δεδομένων.

Ο κώδικας της κλάσης SaferFrame βρίσκεται στο ΠΑΡΑΡΤΗΜΑ.

3.4 Συλλογή και αποστολή δεδομένων

Προτού γίνει ενεργοποίηση της καταγραφής η συσκευή θα πρέπει να έχει τοποθετηθεί σε ένα σταθερό σημείο κοντά στο κέντρο βάρους του οχήματος. Έτσι ακριβώς έγιναν και οι καταγραφές κατά την πειραματική λειτουργία και δοκιμή της εφαρμογής. Από τη στιγμή που αρχίσει η καταγραφή ο χρήστης δε πρέπει να κάνει τίποτε άλλο παρά να αφήσει τη συσκευή σταθερή στη βάση της.

Εφόσον ο χρήστης πατήσει το κουμπί καταγραφής στο επάνω αριστερά μέρος της εφαρμογής η αρχική οθόνη θα είναι όπως η Εικόνα 8. Στη συγκεκριμένη εικόνα παρατηρούμε ότι το κουμπί έναρξης καταγραφής έχει αλλάξει χρώμα και σχήμα και οι τιμές ταχύτητας, ακρίβειας GPS και επιταχύνσεων έχουν ενεργοποιηθεί. Έτσι γίνεται κατανοητό ότι η εφαρμογή βρίσκεται σε κατάσταση καταγραφής. Η διαδικασία είναι αυτόματη και θα σταματήσει μόνο εφόσον πατηθεί εκ νέου το κουμπί πάνω αριστερά.



Εικόνα 8 - Εφαρμογή Safe-R σε ώρα καταγραφής

Όπως προαναφέρθηκε στην περίοδο καταγραφής δημιουργούνται Snapshots τα οποία συγκεντρώνονται σε ένα Frame. Όταν λοιπόν πατηθεί το κουμπί καταγραφής ώστε να σταματήσει η καταγραφή τότε επιχειρείτε να γίνει αποστολή των δεδομένων στο server (Safe House).

3.4.1 Μορφοποίηση JSON

Έχουμε, λοιπόν, κάποια δεδομένα τα οποία έχουν συλλεχθεί από την εφαρμογή μας και είναι έτοιμα προς αποστολή. Η μορφή με την οποία θα σταλούν είναι αρχεία τύπου JSON. Το JSON αποτελεί μία μορφή αρχείων κειμένου τα οποία είναι εύκολο να διαβαστούν και από ανθρώπους και είναι ανεξάρτητα των γλωσσών προγραμματισμού. Μπορούν να δημιουργηθούν αλλά και να διαβαστούν από σχεδόν όλες τις γνωστές γλώσσες προγραμματισμού. Έχουν, μάλιστα, δημιουργηθεί και πολλές έτοιμες βιβλιοθήκες για πολλές γλώσσες προγραμματισμού έτσι ώστε η διαδικασία μετατροπής σε JSON και αντίστροφα να αυτοματοποιείται σε μεγάλο βαθμό.

Τα δεδομένα της δειγματοληψίας (Snapshots) έχουν ομαδοποιηθεί σε ένα Frame. Αυτό το Frame μετατρέπουμε σε μορφή JSON με τη βοήθεια μίας βιβλιοθήκης που παρέχεται από την Google και ονομάζεται GSON. Η μετατροπή των δεδομένων σε JSON γίνεται με χρήση της παρακάτω μεθόδου (**toMap**), η οποία βρίσκεται στην κλάση SaferFrame.

```
public Map<String, String> toMap() {
    final Map<String, String> framesHashMap = new HashMap<>();
    Gson gson = new GsonBuilder().setVersion(0.1).setPrettyPrinting().create();
    framesHashMap.put("data", gson.toJson(this));
    return framesHashMap;
}
```

Και ένα παράδειγμα ενός JSON αρχείου που αποστέλλεται στο server είναι το εξής:

```
{
  "data": {
    "frame": {
      "userID": 1,
      "deviceID": 1,
      "vehicleID": 1,
      "snapshotsAmount": 2
    },
    "snapshots": [
      {
        "avgForce": "11.2314",
        "maxForce": "12.8673",
        "varForce": "5.634",
        "speed": "45.2",
        "lat": "21.12312411",
        "lon": "38.11251213",
        "accuracy": "4"
      },
      {
        "avgForce": "9.924",
        "maxForce": "10.343",
        "varForce": "2.132",
        "speed": "44.4",
        "lat": "21.1232521",
        "lon": "38.113421",
        "accuracy": "4"
      }
    ]
  }
}
```

Είναι ευδιάκριτα τα βασικά στοιχεία που δομούν το αρχείο JSON των δειγματοληψιών. Στο συγκεκριμένο παράδειγμα έγινε συλλογή δεδομένων για μόλις δύο δευτερόλεπτα. Τα δεδομένα "data" χωρίζονται σε "frame" και "snapshots". Στο "frame" πεδίο βρίσκονται οι πληροφορίες που χαρακτηρίζουν όλα τα Snapshots, δηλαδή το Frame. Αντίστοιχα το πεδίο "snapshots" έχει τα δύο Snapshots με όλα τα στοιχεία τους σε μία λίστα. Παρατηρούμε πως το πεδίο snapshotsAmount έχει τιμή δύο, όσα δηλαδή και τα snapshots.

Όταν το JSON αυτό αρχείο σταλεί επιτυχώς στο server θα αποκωδικοποιηθούν τα snapshots που εμπεριέχονται και θα μετατραπούν σε νέες εγγραφές στη Βάση Δεδομένων. Υπάρχει επομένως συσχέτιση της δομής του JSON και της μορφής της Βάσης Δεδομένων. Περισσότερο εκτενής αναφορά στις διαδικασίες του server θα γίνει στο επόμενο κεφάλαιο.

3.4.2 Ασφάλεια αποστολής - OAuth 2.0

Τίθεται ένα θέμα στο ποιός μπορεί να στείλει τέτοια δεδομένα στο server και εάν θα έπρεπε να γίνονται δεκτά δεδομένα από κάθε πηγή. Εφόσον θέλουμε να μπορεί μόνο η εφαρμογή να στέλνει τέτοια JSON αρχεία στο server έπρεπε να γίνει μία υλοποίηση ενός υποσυστήματος αυθεντικοποίησης της εφαρμογής ώστε στη συνέχεια να γίνονται αποδεκτά τα δείγματα που στέλνει. Θα έπρεπε να υπάρχει ένας τρόπος ελέγχου της ταυτότητας του αποστολέα ώστε να μη γίνονται αποδεκτά παρόμοιας μορφής δεδομένα από μη έγκυρες πηγές.

Επιλέχθηκε η αρχιτεκτονική του OAuth 2.0 ώστε να γίνει η αυθεντικοποίηση της εφαρμογής. Η λογική πίσω από το OAuth είναι ότι για να επιτευχθεί η αποστολή

δεδομένων θα πρέπει να αποστέλλεται μαζί με αυτά ένα access token το οποίο έχει χρονικό όριο ζωής και μόνο έτσι θα επιτρέπεται στην εφαρμογή να στέλνει δεδομένα.

Η διαδικασία έκδοσης τέτοιων access tokens έχει κάποια συγκεκριμένα βήματα και στάδια ώστε να διασφαλίζεται η ταυτότητα του χρήστη. Όπου χρήστης στη δική μας περίπτωση θεωρείται η εφαρμογή.

Αρχικά, αποστέλλεται ένα αίτημα δημιουργίας access token στο server με τα εξής στοιχεία:

- client id
- client secret
- username
- password

Αυτά τα τέσσερα στοιχεία ελέγχονται από το server και εάν είναι έγκυρα και σχετίζονται μεταξύ τους δημιουργείται ένας προσωρινός κωδικός (access token) και αποστέλλεται πίσω στην εφαρμογή. Η εφαρμογή έχει ένα γνωστό client id και client secret. Αυτά τα δύο πεδία τα γνωρίζει και ο server. Το μόνο κρυφό πεδίο στο server είναι αυτό του κωδικού (password). Υπάρχει μόνο η υπογραφή του αποθηκευμένη στη βάση δεδομένων. Όπως αυτή προκύπτει μετά την εισαγωγή του κωδικού σε μία συνάρτηση κατακερματισμού και πιο συγκεκριμένα τη συνάρτηση SHA-1.

Επομένως ο server καλείται να συγκρίνει την υπογραφή που έχει αποθηκευμένη με την υπογραφή του κωδικού που αποστέλλεται και τέλος να βεβαιώσει ότι ο κωδικός αυτός αντιστοιχεί στο συγκεκριμένο client. Μόνο τότε θα σταλεί ένα access token το οποίο θα έχει διάρκεια ζωής 24 ώρες. Στη συνέχεια αυτό το token απλά εισάγεται στην κεφαλίδα (Header) των αιτήσεων που στέλνει η εφαρμογή και έτσι γίνονται αποδεκτά τα δεδομένα από τον server. Σε αντίθετη περίπτωση επιστρέφεται απάντηση λάθους από το server.

3.5 Τοπική αποθήκευση δεδομένων

Οι κινητές συσκευές τύπου Smartphone επιτρέπουν τη σύνδεση στο Internet είτε μέσω των δικτύων κινητής τηλεφωνίας είτε μέσω τοπικών δικτύων Wi-Fi. Η δυνατότητα αυτή όμως υπάρχει περίπτωση να μη παρέχεται κάθε στιγμή. Είτε λόγω απουσίας δικτύων Wi-Fi είτε λόγω κακού σήματος των δικτύων της κινητής είτε απλά επειδή ο χρήστης έχει απενεργοποιημένες αυτές τις υπηρεσίες.

Κατά τη λειτουργίας της, η εφαρμογή, έχει ως βασική προϋπόθεση τη λειτουργία του GPS. Παρόλα αυτά και επειδή πρέπει να αποσταλούν τα δεδομένα που συλλέγονται στο server υπάρχει και η ανάγκη χρήσης του Internet ώστε να γίνει αυτή η επικοινωνία. Οπότε, όταν για κάποιο λόγο δεν υπάρχει σύνδεση με το Internet γίνεται επιτακτική η ανάγκη τοπικής αποθήκευσης των δεδομένων που έχουν συλλεχθεί και η αποστολή τους αργότερα.

Δημιουργήθηκε, λοιπόν, μία κλάση, η οποία δίνει τη δυνατότητα τοπικής αποθήκευσης των δεδομένων στην εφαρμογή, με το όνομα **SaferDatabaseHandler**.

3.5.1 Η κλάση SaferDatabaseHandler

Η κλάση **SaferDatabaseHandler** αποτελεί επέκταση της κλάσης **SQLiteOpenHelper** και διαχειρίζεται εξ ολοκλήρου μία τοπική βάση δεδομένων τύπου SQLite.

Η SQLite αποτελεί μία σχεσιακή βάση δεδομένων όπως και η MySQL μόνο που δεν υπάρχει η απαίτηση κάποιου εξυπηρετητή ώστε να εγκατασταθεί παρά μόνο υπάρχει μαζί με την εφαρμογή στον πελάτη και έχει τοπική ισχύ. Όλα η βάση δεδομένων όπως και τα δεδομένα που χειρίζεται βρίσκονται σε ένα και μοναδικό αρχείο και δεν υπάρχει κάποιου είδους έλεγχος πρόσβασης σε αυτή. Ο μοναδικός έλεγχος πρόσβασης που υπάρχει είναι αυτός που εφαρμόζεται στους ιδιοκτήτες των αρχείων από το λειτουργικό σύστημα (στην περίπτωσή μας το Android). Εφόσον η βάση δεδομένων δημιουργείται από την εφαρμογή Safe-R αυτή η βάση έχει ως ιδιοκτήτη την εφαρμογή και μόνο η εφαρμογή έχει πρόσβαση σε αυτή.

Η μέθοδος **onCreate** της κλάσης **SaferDatabaseHandler** δημιουργεί το μόνο πίνακα που θα χρειαστεί η εφαρμογή. Τον πίνακα "**offlineFrames**". Ο πίνακας αυτός, όπως υποδηλώνει και το όνομά του, αποθηκεύει τα Frames τα οποία καταγράφηκαν όσο η εφαρμογή δεν είχε πρόσβαση στο διαδίκτυο. Ο κώδικας της μεθόδου onCreate είναι ο εξής:

```
public void onCreate(SQLiteDatabase db) {
    String CREATE_CONTACTS_TABLE = "CREATE TABLE " + TABLE_OFFLINE_FRAMES + "("
            + TABLE_OFFLINE_FRAMES_ID_COLUMN + " INTEGER PRIMARY KEY AUTOINCREMENT,"
            + TABLE_OFFLINE_FRAMES_OFFLINE_FRAME_COLUMN + " BLOB NOT NULL,"
            + TABLE_OFFLINE_FRAMES_DATE_CREATED_COLUMN + " INT NOT NULL)";
    db.execSQL(CREATE_CONTACTS_TABLE);
}
```

Το συντακτικό των ερωτημάτων SQL ακολουθεί τη λογική άλλων βάσεων δεδομένων (PostgreSQL, MySQL, κλπ.). Ο πίνακας **offlineFrames** που δημιουργείται από τη μέθοδο onCreate έχει τρεις στήλες. Το κυρίως κλειδί (id), την ημερομηνία δημιουργίας της εγγραφής (dateCreated) και τέλος μία στήλη με το όνομα offlineFrame η οποία είναι τύπου BLOB και έχει τη δυνατότητα αποθήκευσης δυαδικών αρχείων. Στην περίπτωση της εφαρμογής Safe-R τα αρχεία αυτά θα είναι τα αρχεία τύπου JSON τα οποία είναι προς αποστολή στο server.

Μία ακόμη σημαντική μέθοδος της κλάσης αποτελεί η μέθοδος **addOfflineData** η οποία επιτρέπει σε ένα πίνακα με τιμές να μετατραπεί σε δυαδικής μορφής πίνακα και στη συνέχεια να εισαχθεί στη τοπική βάση δεδομένων στον πίνακα που η onCreate

δημιούργησε. Όπως φαίνεται στον κώδικα παρακάτω αρχικά γίνεται η μετατροπή του πίνακα με τα δείγματα (Snapshots) σε δυαδικό πίνακα με τη μέθοδο **getByteArrayFromParams** και στη συνέχεια δημιουργείται μία νέα εγγραφή στον πίνακα offlineFrames μαζί με την ώρα καταγραφής.

```
public void addOfflineData(Map<String, String> params) {
    SQLiteDatabase db = this.getWritableDatabase();

    byte[] byteArray = getByteArrayFromParams(params);// Convert Map to byte array
    Long tsLong = System.currentTimeMillis() / 1000;
    String currentTimestamp = tsLong.toString();

    ContentValues values = new ContentValues();
    values.put(TABLE_OFFLINE_FRAMES_OFFLINE_FRAME_COLUMN, byteArray);
    values.put(TABLE_OFFLINE_FRAMES_DATE_CREATED_COLUMN, currentTimestamp);

    db.insert(TABLE_OFFLINE_FRAMES, null, values);
    db.close();

    Safer.longToast("Frame stored locally...\n(" + getUnsentFramesCount() + " Frames in
SQLite)");
}

private byte[] getByteArrayFromParams(Map<String, String> params) {
    ByteArrayOutputStream byteOut = new ByteArrayOutputStream();
    try {
        ObjectOutputStream out = new ObjectOutputStream(byteOut);
        out.writeObject(params);
    } catch (IOException e) {
        e.printStackTrace();
    }
    return byteOut.toByteArray();
}
```

Τέλος, ενημερώνεται ο χρήστης για την τοπική αποθήκευση των δεδομένων με το αντίστοιχο μήνυμα.

Υπάρχει και η αντίστοιχη μέθοδος για την εξαγωγή των δεδομένων από τον πίνακα η οποία ονομάζεται **getAllUnsentFrames**. Η συγκεκριμένη μέθοδος εκτελεί την ακριβώς αντίστροφη διαδικασία από την **addOfflineData**. Βρίσκει όλες τις εγγραφές του πίνακα offlineFrames και τις μετατρέπει σε πίνακα δεδομένων ώστε στη συνέχεια να γίνουν JSON και να αποσταλούν στο server.

Ο κώδικας της κλάσης SaferDatabaseHandler βρίσκεται στο ΠΑΡΑΡΤΗΜΑ.

3.6 Αναπαράσταση δεδομένων σε χάρτη

Μέχρι στιγμής έχουμε αναφερθεί μόνο για τη διαδικασία συλλογής των δεδομένων ποιότητας του οδοστρώματος και ελάχιστα για τα υπόλοιπα μέρη της διαδικασίας επεξεργασίας και αναπαράστασής τους στην εφαρμογή. Σε αυτό το υποκεφάλαιο θα γίνει μία αρχική ενημέρωση για την επιλογή του χάρτη της εφαρμογής, στη συνέχεια θα περιγραφεί ο τρόπος με τον οποίο σχεδιάζεται ο χάρτης ώστε να γίνει αναπαράσταση της ποιότητας των δρόμων και τέλος δίνονται κάποιες λεπτομέρειες βελτιστοποίησης των διαδικασιών που σχετίζονται με το χάρτη.

3.6.1 Επιλογή Χάρτη

Η κυριότερη επιλογή όταν αναφερόμαστε σε ηλεκτρονικούς χάρτες αποτελούν οι χάρτες της Google και το Google Maps Android API αποτελεί τη διεπαφή προγραμματισμού έτσι ώστε να μπορέσει κάποιος χρήστης να εισάγει δεδομένα και να χειριστεί αυτούς τους χάρτες σε μία εφαρμογή.

Τα εργαλεία και οι αυτοματισμοί που παρέχονται από την Google είναι πολύ βοηθητικά για τη δημιουργία απλών εφαρμογών που εμπεριέχουν χάρτες. Δυστυχώς όμως υπάρχουν κάποιοι περιορισμοί σε αυτά τα δεδομένα που θα μπορούσαν να χρήσουν την εφαρμογή πιο δύσχρηστη. Ένα παράδειγμα είναι το όριο σε αναζητήσεις στο χάρτη από κάθε εφαρμογή. Αυτό έχει οριστεί στις 25.000 αναζητήσεις ανά ημέρα και ανά εφαρμογή. Κάτι τέτοιο σημαίνει πως εάν η εφαρμογή είχε αρκετούς χρήστες θα έπρεπε να υπάρχει πληρωμή για να επεκταθεί η χρήση πέραν των 25.000 αναζητήσεων. Επίσης τα δεδομένα που παρέχει η Google όπως και οι δυνατότητες αναπαράστασης επάνω στο χάρτη είναι ορισμένες και περιορίζουν τον προγραμματισμό πιο περίπλοκων σχημάτων και σχεδίων επάνω στους χάρτες της. Και τέλος η Google περιορίζει αρκετά τη δημιουργία εξειδικευμένων χαρτών με χρήση των γεωχωρικών δεδομένων της. Ακόμη και εάν η εφαρμογή Safe-R δεν έχει δημιουργήσει ένα νέο είδος χάρτη κάτι τέτοιο θα ήταν απαγορευμένο να γίνει στο μέλλον με τη χρήση των χαρτών της Google.

Επομένως, έχοντας τα προηγούμενα εμπόδια δεν έγινε η επιλογή των χαρτών της Google. Μετά από έρευνα για αντίστοιχους ηλεκτρονικούς χάρτες επιλέχθηκαν οι χάρτες του **OpenStreetMap (OSM)**. Όλα τα προηγούμενα προβλήματα δεν ισχύουν με τους OSM χάρτες και δίνεται έτσι ακόμα και δυνατότητα δημιουργίας νέου τύπου χάρτη στο μέλλον, ο οποίος θα εμπεριέχει τα δεδομένα αναταραχών ενσωματωμένα σε αυτόν.

Ο χάρτης του OpenStreetMap αποτελεί ένα χάρτη με ελεύθερη άδεια χρήσης ο οποίος αναπτύσσεται από μία κοινότητα εθελοντών (ιδιωτών ή εταιρειών) που συνεισφέρουν και διατηρούν δεδομένα σχετικά με δρόμους, μονοπάτια, σιδηροδρομικούς σταθμούς και πολλά περισσότερα, σε όλον τον κόσμο. Οι συνεισφέροντες χρησιμοποιούν αεροφωτογραφίες, συσκευές GPS και τοπικούς χάρτες ώστε να σιγουρευτούν ότι το OSM είναι ακριβές και ενημερωμένο. Μέχρι το 2012 είχαν συνεισφέρει στη δημιουργία του OSM χάρτη πάνω από 500.000 άνθρωποι.

Όλα τα δεδομένα του OSM διατίθενται ελεύθερα με άδεια Open Data Commons Database License (ODbL) τα οποία μπορεί να χρησιμοποιεί κανείς για οποιονδήποτε σκοπό, εφόσον μνημονευθεί το OpenStreetMap και οι συνεισφέροντες του. Η φιλοξενία των γεωχωρικών δεδομένων που απαιτούνται από όποια εφαρμογή επιλέγει τη χρήση των OSM χαρτών υποστηρίζονται από το UCL VR Centre στο Imperial College του Λονδίνου, την Bytemark Hosting και άλλους. Το OpenStreetMap αποτελεί πρωτοβουλία του Στηβ Κόουστ (Steve Coast) και δημιουργήθηκε το 2004.

3.6.2 Οι βιβλιοθήκες osmdroid & osmbonuspack

Η βασικότερη βιβλιοθήκη που χρησιμοποιήθηκε ώστε να δημιουργηθεί ο ηλεκτρονικός OSM χάρτης είναι η βιβλιοθήκη **osmdroid**. Μαζί με κάποιες πολύ χρήσιμες μεθόδους και κλάσεις της βιβλιοθήκης **osmbonuspack** αποτέλεσαν τα βασικά εργαλεία αρχικοποίησης και χειρισμού του χάρτη στην εφαρμογή Safe-R.

Σύμφωνα με τη σελίδα www.osmdroid.org το osmdroid αποτελεί μία (σχεδόν) πλήρη/δωρεάν αντικατάσταση της κλάσης MapView, η οποία είναι η βασική κλάση με την οποία δημιουργούνται και διαχειρίζονται οι χάρτες της Google. Επομένως, η χρήση της βιβλιοθήκης osmdroid μας παρέχει δυνατότητες σχεδόν παρόμοιες με αυτές που παρέχει η Google για τους χάρτες της. Για την ακρίβεια το project osmdroid είναι μία ανοιχτού κώδικα πρωτοβουλία της ίδιας της Google.

Λίγο αργότερα από τη δημιουργία του osmdroid και επειδή οι ανάγκες αυτοματισμών ήταν αρκετές δημιουργήθηκε και η βιβλιοθήκη osmbonuspack. Σύμφωνα με τη σελίδα του project στο Github (http://github.com/MKergall/osmbonuspack) η βιβλιοθήκη osmbonuspack έχει τη δυνατότητα να επιδρά με δεδομένα των OSM χαρτών στα πλαίσια μίας εφαρμογής Android. Προσφέρει ένα υποκατάστατο για πολύ βασικά μοντέλα χωρικών δεδομένων που παρέχει και η Google στους χάρτες της. Βασικά χωρικά μοντέλα αποτελούν το σημείο (Point), το πολύγωνο (Polygon), η γραμμή (Polyline) κ.α.. Έχει σχεδιαστεί ώστε να χρησιμοποιείται επικουρικά με την βιβλιοθήκη osmdroid.

3.6.3 Δημιουργία του χάρτη

Έτσι ώστε να γίνει εφικτή η δημιουργία αλλά και ο χειρισμός του χάρτη δημιουργήθηκε μία κλάση με όνομα **SaferMapManager**. Σε αυτή την κλάση γίνεται και η αρχικοποίηση των βασικών ρυθμίσεων του χάρτη. Ορίζονται οι εξής παράμετροι στο χάρτη:

- Ενεργοποιείται η λειτουργία πολλαπλής αφής (multi touch control)
- Τίθεται το μέγιστο/ελάχιστο επιτρεπόμενο επίπεδο εστίασης του χάρτη
- Ορίζεται ένα αρχικό σημείο προβολής για το χάρτη
- Ορίζεται ένα αρχικό επίπεδο εστίασης του χάρτη
- Ενεργοποιείται η προβολή της τωρινής θέσης της συσκευής
- Δημιουργείται ένα πολύγονο το οποίο ορίζει την περιοχή δοκιμών, όπως αυτή φαίνεται στο χάρτη της Εικόνα 2
- Ανατίθεται μία κλάση ώστε να παρακολουθεί και να αντιδρά σε όλες τις δράσεις που γίνονται στο χάρτη

Η δημιουργία του πολυγώνου γίνεται έτσι ώστε να γίνεται έλεγχος στα νέα δείγματα που συλλέγονται. Θα πρέπει όλα τα δείγματα να ανήκουν στο χώρο δοκιμών που έχει οριστεί για την εφαρμογή. Σε διαφορετική περίπτωση αυτά θα απορρίπτονται.

Η κλάση που ανατίθεται ώστε να παρακολουθεί τις δράσεις που γίνονται στο χάρτη ονομάζεται **SaferMapListener** και θα περιγραφεί στην επόμενη ενότητα με λεπτομέρεια καθότι εμπεριέχει πολύ σημαντικές λειτουργίες του χάρτη.

Ο κώδικας της κλάσης SaferMapManager βρίσκεται στο ΠΑΡΑΡΤΗΜΑ.

3.6.4 Εισαγωγή δρόμων στο χάρτη

Όπως αναφέρθηκε και στη προηγούμενη ενότητα κατά τη δημιουργία του χάρτη γίνεται και μία ανάθεση μίας κλάσης - παρατηρητή των γεγονότων και των συμβάντων που συμβαίνουν στο χάρτη. Αυτή η κλάση ονομάζεται **SaferMapListener** και είναι υπεύθυνη

για τη συλλογή των επεξεργασμένων δρόμων από το server και την απεικόνισή τους στο χάρτη.

Οι δύο βασικότερες μέθοδοι της κλάσης SaferMapListener είναι η **onScroll** και η **onZoom**. Και οι δύο αυτές μέθοδοι καλούνται αυτόματα όταν συμβαίνουν δύο συγκεκριμένα γεγονότα στο χάρτη, όταν ο χρήστης κινεί το ορατό μέρος του χάρτη (onScroll) και όταν ο χρήστης αλλάζει το βαθμό εστίασης του χάρτη (onZoom). Κάθε φορά που μία από τις δύο αυτές μεθόδους καλούνται ενεργοποιείται ένας συγκεκριμένος μηχανισμός ανανέωσης του χάρτη με δεδομένα από το server. Ας μελετήσουμε ξεχωριστά, όμως, τις δύο αυτές μεθόδους καθώς χειρίζονται τελείως διαφορετικά γεγονότα.

Αρχικά έχουμε την **onScroll**. Η παραμικρή κίνηση του χάρτη θα προκαλέσει την κλήση της συγκεκριμένης μεθόδου. Οπότε αντιλαμβάνεται κανείς πως αυτή η μέθοδος θα κλιθεί δεκάδες φορές σε μία και μόνο κίνηση του χάρτη. Αυτό θα μπορούσε να προκαλέσει την αποστολή πάρα πολλών αιτημάτων στο server για ανανέωση του χάρτη. Κάτι τέτοιο δεν επιθυμητό καθώς δε θέλουμε να υπάρχουν πάρα πολλά αιτήματα στο server για λόγους απόδοσής του. Για να περιοριστεί η αποστολή πολλών αιτημάτων ελέγχονται οι εξής παράμετροι κάθε φορά που η μέθοδος onScroll καλείται:

- Έχει απαντηθεί από το server το προηγούμενο αίτημα ανανέωσης επιτυχώς.
- Έχουν περάσει τουλάχιστον δύο δευτερόλεπτα από το τελευταίο αίτημα για ανανέωση που στάλθηκε από την εφαρμογή.
- Έχει κινηθεί ο χάρτης τουλάχιστον 50 μέτρα από το προηγούμενο σημείο ανανέωσης του χάρτη.

Θα πρέπει και οι τρεις προηγούμενες συνθήκες να έχουν ικανοποιηθεί ώστε να πραγματοποιηθεί εκ νέου αίτημα ανανέωσης του χάρτη.

Από την άλλη έχουμε τη μέθοδο **onZoom**. Αυτή η μέθοδος καλείται με κάθε αλλαγή του επίπεδου εστίασης του χάρτη. Όπως και η onScroll έτσι και η συγκεκριμένη μέθοδος καλείται αρκετές φορές σε κάθε αλλαγή εστίασης οπότε ορίστηκαν συγκεκριμένες συνθήκες ώστε να γίνεται προσπάθεια ανανέωσης του χάρτη. Οι παράμετροι που ορίστηκαν είναι οι εξής:

- Έχει απαντηθεί από το server το προηγούμενο αίτημα ανανέωσης επιτυχώς.
- Έχουν περάσει τουλάχιστον δύο δευτερόλεπτα από το τελευταίο αίτημα για ανανέωση που στάλθηκε από την εφαρμογή.
- Το επίπεδο εστίασης του προηγούμενου αιτήματος ανανέωσης ήταν μεγαλύτερο από το νέο επίπεδο εστίασης. Δηλαδή, ο χρήστης έχει εμφανίσει μεγαλύτερο μέρος του χάρτη.

Θα πρέπει και οι τρεις προηγούμενοι παράμετροι να ισχύουν ώστε να πραγματοποιηθεί ένα νέο αίτημα ανανέωσης του χάρτη.

Αυτή η διαδικασία σταδιακής και ελεγχόμενης ανανέωσης κρατάει σε χαμηλά επίπεδα τα αιτήματα στο server και κρατάει ενημερωμένο σε ικανοποιητικό βαθμό το χάρτη χωρίς να υπάρχουν ελλιπή ή καθόλου δεδομένα για δρόμους που έχουν εξεταστεί από την εφαρμογή.

Οι μέθοδοι onScroll και onZoom, όπου αναλύθηκαν προηγουμένως καλούν μία συγκεκριμένη μέθοδο, η οποία αναλαμβάνει την αποστολή αιτήματος στο server για ανανέωση του χάρτη και αργότερα τη συλλογή της απάντησης του server και της ενημέρωσής του. Αυτή η μέθοδος ονομάζεται **fetchRoadDataForViewableArea**.

Η μέθοδος **fetchRoadDataForViewableArea** στην ουσία δε στέλνει κάποιο αρχείο τύπου JSON στο server παρά μόνο προσπελαύνει μία συγκεκριμένη διεύθυνση δικτύου στην οποία ορίζονται οι παράμετροι θέσης του χάρτη και ως απάντηση λαμβάνει ένα αρχείου τύπου JSON με τα δεδομένα των δρόμων και των αναταραχών τους.

Το URL το οποίο κατασκευάζεται από τη μέθοδο αυτή έχει την εξής μορφή:

```
http://www.safe-r.gr/api/v1/roads/northWestLon/northWestLat/southEastLon/southEastLat
```

Όπου:

- Το *http://www.safe-r.gr/api/v1/roads* αποτελεί το βασικό μονοπάτι της διεύθυνσης URL.
- Το **northWestLon** είναι το γεωγραφικό μήκος του βορειοδυτικού άκρου του χάρτη κατά την αποστολή του αιτήματος.
- Το **northWestLat** είναι το γεωγραφικό πλάτος του βορειοδυτικού άκρου του χάρτη κατά την αποστολή του αιτήματος.
- Το **southEastLon** είναι το γεωγραφικό μήκος του νοτιοανατολικού άκρου του χάρτη κατά την αποστολή του αιτήματος.
- Το **southEastLat** είναι το γεωγραφικό πλάτος του νοτιοανατολικού άκρου του χάρτη κατά την αποστολή του αιτήματος.

Η απάντηση, σε μορφή JSON, περιλαμβάνει μία λίστα με τα εξής στοιχεία για κάθε δρόμο που περιλαμβάνεται από το παραλληλόγραμμο που ορίζεται μεταξύ των προαναφερθέντων γεωγραφικών στοιχείων. Ακολουθεί παράδειγμα απάντησης με ένα μόνο δρόμο σε αυτή.


```
{
  "data": [
    {
      "gid": 32965,
      "averageForce": 9.85,
      "maxForce": 11.21,
      "varForce": 0.62,
      "averageSpeed": 27.96,
      "updateCounter": 2,
      "points": [
        [
          23.7669079,
          37.9852547
        ],
        [
          23.7668716,
          37.9855327
        ],
        [
          23.7667514,
          37.9856409
        ],
        [
          23.7665359,
          37.9858007
        ],
        [
          23.7661959,
          37.9860422
        ],
        [
          23.7658047,
          37.9863535
        ],
        [
          23.765708,
          37.9864305
        ]
      ]
    },
```

```
"error": {
    "description": "",
    "code": ""
  }
}
```

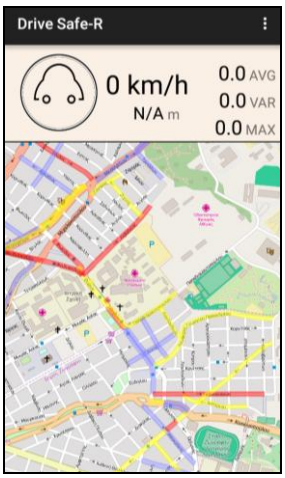
Στο JSON παρατηρούμε τα εξής δεδομένα:

- **gid:** Αποτελεί ένα μοναδικό αριθμό που αντιστοιχεί με ένα και μόνο δρόμο στη βάση δεδομένων.
- **averageForce:** Ο μέσος όρος των επιταχύνσεων που μετρήθηκαν στο μήκος του συγκεκριμένου δρόμου.
- **maxForce:** Ο μέσος όρος των μέγιστων επιταχύνσεων που μετρήθηκαν στο μήκος του συγκεκριμένου δρόμου.
- **varForce:** Ο μέσος όρος των διασπορών στις μετρήσεις της επιτάχυνσης που μετρήθηκαν στο μήκος του συγκεκριμένου δρόμου.
- **averageSpeed:** Η μέση ταχύτητα που αναπτύχθηκε κατά τη δειγματοληψία στο συγκεκριμένο δρόμο.
- **updateCounter:** Το σύνολο των Snapshot που έχουν χρησιμοποιηθεί ώστε να εκτιμηθεί η κατάσταση του συγκεκριμένου δρόμου.
- **points:** Τα σημεία (γεωγραφικό μήκος/πλάτος) που ορίζουν το δρόμο.

Μετά τη μετατροπή του αρχείου JSON σε αντικείμενα τύπου **SaferRoad** με τη βοήθεια της βιβλιοθήκης GSON γίνεται σχεδιασμός σημείο προς σημείο των δρόμων με χρήση μίας συνάρτησης που ανήκει στο πακέτο μεθόδων του osmdroid. Εκτός από τα σημεία του δρόμου, σε μορφή Polyline (ακολουθία σημείων), απαιτείται και ένα χρώμα για το σχεδιασμό του δρόμου κατάλληλα στο χάρτη. Το χρώμα στον κάθε δρόμο αποφασίζεται από τη μέθοδο **getColor** που ανήκει στην κλάση SaferRoad, δηλαδή κάθε δρόμος αντιστοιχίζεται σε ένα αντικείμενο της κλάσης SaferRoad και έχει ένα συγκεκριμένο χρώμα.

Ο τρόπος με τον οποίο προκύπτει το χρώμα για κάθε δρόμο προέκυψε μετά από αρκετές δοκιμές της ευαισθησίας του επιταχυνσιόμετρου της συσκευής. Ορίστηκαν συγκεκριμένες στάθμες κατωφλίου για κάθε ένα από τα τρία πιθανά επίπεδα ποιότητας του οδοστρώματος για τις μετρήσεις της μέσης και μέγιστης τιμής επιτάχυνσης.

Έτσι έχοντας συλλέξει μετρήσεις για τη μέση και μέγιστη επιτάχυνση στα σημεία του κάθε δρόμου μία απλή συνάρτηση κατωφλίου ορίζει την κατηγορία του κάθε δρόμου και επιλέγεται ένα από τα τρία βασικά χρώματα μπλε, κίτρινο και κόκκινο ώστε να αναπαραστήσουν την καλή, μέτρια και κακή ποιότητα δρόμου αντίστοιχα. Αποτέλεσμα της διαδικασίας εισαγωγής φαίνεται στην Εικόνα 9.



Εικόνα 9 - Ενημερωμένος Χάρτης

3.6.5 Απομνημόνευση δρόμων

Όσο ο χρήστης κινείται στο χάρτη τόσο πραγματοποιούνται αιτήματα ανανέωσής του. Αποτέλεσμα είναι να γίνεται υπερφόρτωση της συσκευής με δεδομένα που μπορεί να αφορούν πολλές φορές τους ίδιους ακριβώς δρόμους.

Οπότε κρίθηκε αναγκαίο να υπάρχει μία είδους απομνημόνευση των δρόμων οι οποίοι έχουν ήδη σχεδιαστεί επάνω στο χάρτη. Το gid του κάθε δρόμου ήταν το ιδανικό ώστε να χαρακτηρίσει τον κάθε δρόμο μοναδικά στη μνήμη της εφαρμογής. Κάθε νέος δρόμος που εισάγεται προσθέτει το δικό του gid στη μνήμη και σε μελλοντικές ανανεώσεις του χάρτη δεν επανεισάγεται στο χάρτη.

Βεβαίως εφόσον αναφερόμαστε σε μία εφαρμογή διαδραστική δε πρέπει να έχουμε αποθηκευμένους μόνιμα τους δρόμους στη μνήμη και να μη γίνεται ποτέ ανανέωση τους από το server καθώς υπάρχει η πιθανότητα να έχουν γίνει νέες μετρήσεις για αυτούς. Ορίστηκε, λοιπόν, ένα όριο ζωής για τα gid που κρατούνται στη μνήμη. Αυτό το όριο είναι η μία ώρα. Έπειτα από μία ώρα όλοι οι δρόμοι πρέπει να ανανεωθούν και πάλι από το server.

Τέλος, δεν αρκεί μόνο η προσωρινή αποθήκευση των δρόμων ώστε να περιοριστεί η κίνηση που προκαλείται στο server από τα αλλεπάλληλα αιτήματα της εφαρμογής για ανανέωση των δρόμων. Έγινε μία προσπάθεια περαιτέρω περιορισμού των αιτημάτων που αποστέλλονται στο server με την εξής λογική ελέγχου.

Σε κάθε ανανέωση που πραγματοποιείται αποθηκεύονται στη μνήμη τα στοιχεία του χάρτη εκείνης της χρονικής στιγμής. Αποθηκεύεται το γεωγραφικό μήκος και πλάτος του βορειοδυτικού και νοτιοανατολικού άκρου του χάρτη. Έτσι γνωρίζουμε ανά πάσα στιγμή τις περιοχές στις οποίες έχει γίνει επιτυχής ανανέωση του χάρτη. Όποτε ο χάρτης βρεθεί σε περιοχή εντός μίας εκ των αποθηκευμένων περιοχών τότε δε γίνεται εκ νέου ανανέωση του χάρτη. Η μνήμη αυτών των περιοχών έχει ακριβώς τον ίδιο μηχανισμό καθαρισμού ανά μία ώρα όπως και η λίστα με τα gid των δρόμων που περιγράφηκε προηγουμένως. Έτσι σε περίπτωση που έχουν εισαχθεί νέοι δρόμοι στο σύστημα αυτό γίνεται εμφανές στο χάρτη σε λιγότερο από 60 λεπτά.

Αυτή η απομνημόνευση των περιοχών που έχουν ήδη ανανεωθεί σε συνδυασμό με τις συνθήκες και τους περιορισμούς που ορίσθηκαν για τις μεθόδους onScroll και onZoom μείωσαν δραστικά τα αιτήματα προς το server καθώς επίσης και τη χρήση μνήμης και της απόδοσης της συσκευής. Χρόνο και μνήμη ο οποίος είναι αναγκαίος, στη συσκευή, για τη διαδικασία καταγραφής του οδοστρώματος.

3.7 Προβλήματα - Παραδοχές

3.7.1 Αντικειμενικότητα κατηγοριοποίηση δρόμων

Ένα από τα πιο δύσκολα προβλήματα ήταν η επιλογή των οριακών τιμών έτσι ώστε να υπάρχει σωστή κατηγοριοποίηση των δρόμων ποιοτικά. Εάν επιλέγονταν υψηλά όρια τότε θα οδηγούμασταν σε μία απεικόνιση χάρτη με όλους τους δρόμους χρωματισμένους με πράσινο χρώμα (καλή ποιότητα). Αντιθέτως εάν τα όρια ήταν χαμηλά τότε όλοι οι δρόμοι θα είχαν κόκκινο χρώμα υποδηλώνοντας ότι έχουν όλοι κακή ποιότητα. Επιλέχθηκαν τιμές έπειτα από πειραματική χρήση της εφαρμογής τέτοιες ώστε να ορίζονται ως κακοί δρόμοι εκείνοι που με υποκειμενική κρίση της ομάδας των πειραμάτων θεωρήθηκαν ως κακοί.

Είναι απόλυτα λογικό να μη κρίνει ο καθένας ένα δρόμο ως καλό, μέτριο ή κακό με τον ίδιο τρόπο. Αυτό πολλές φορές μπορεί να οφείλεται και στον τύπο του οχήματος στο με το οποίο κάποιος κινείται σε κάποιο δρόμο. Όμως, έχοντας επιλέξει ένα συγκεκριμένο όχημα για όλη τη διαδικασία της καταγραφής και έχοντας ακριβώς τα ίδια κριτήρια ταξινόμησης των δρόμων μπορούμε να καταλήξουμε σε ένα χάρτη ο οποίος κατηγοριοποιεί σε τρία υποσύνολα τους δρόμους. Αυτή η διάκριση δίνει και την τελική αξία στα αποτελέσματα της δειγματοληψίας. Οδηγούμαστε σε μία ομάδα κακών δρόμων που θα μπορούσαν να αποτελέσουν τους αρχικά επιλεγμένους δρόμους σε μία προσπάθεια ανακατασκευής και βελτίωσης του οδικού δικτύου μίας περιοχής. Οπότε θα μπορούσαμε να πούμε ότι γίνεται μία ταξινόμηση του συνόλου των δρόμων ώστε οι κακοί δρόμοι να έπαιρναν προτεραιότητα σε μία διαδικασία ανακατασκευής.

3.7.2 Βιβλιοθήκες osmdroid και osmbonuspack

Ένα ακόμη πρόβλημα κατά τη διαδικασία της απεικόνισης του χάρτη και των δρόμων σε αυτόν ήταν η αρκετά ελλιπής παροχής οδηγιών και πληροφοριών σχετικά με τη βιβλιοθήκη osmdroid. Σε αντίθεση με τους χάρτες της Google, οι χάρτες του OpenStreetMap αποτελούν μία συλλογική προσπάθεια ανοικτού κώδικα και η βιβλιογραφία σε αυτές τις προσπάθειες συνήθως είναι αποτέλεσμα εθελοντικής εργασίας. Παρόλο που έγινε χρήση και των επιπρόσθετων μεθόδων της βιβλιοθήκης osmbonuspack υπήρξε και πάλι η ανάγκη είτε δημιουργίας νέων μεθόδων είτε χρήση έτοιμων μεθόδων που παρέχει η Google στις δικές τις σχετικές βιβλιοθήκες. Ένα χαρακτηριστικό παράδειγμα ήταν η μέθοδος η οποία ελέγχει εάν ένα δοθέν σημείο εμπεριέχεται σε ένα δοθέν πολύγωνο. Δυστυχώς μία παρόμοια μέθοδος δεν υπήρχε στις βιβλιοθήκες osmdroid και osmbonuspack και έπρεπε να γίνει χρήση της βιβλιοθήκης **PolyUtil** της Google και συγκεκριμένα η μέθοδος **containsLocation**.

3.7.3 Τα γεγονότα του χάρτη

Επίσης οι δύο βασικές μέθοδοι της κλάσης SaferMapListener, onScroll και onZoom, είχαν εκτός της λίγης βιβλιογραφίας και λίγες επιλογές επί του ελέγχου των γεγονότων/συμβάντων που καλούνται να διαχειριστούν. Οπότε έγινε αρκετή προσπάθεια με χρήση αρκετών κριτηρίων ώστε να μη γίνεται υπερβολική αποστολή αιτημάτων στο server.

4. Η Εφαρμογή Επεξεργασίας Δεδομένων, Safe-House

Τα δεδομένα της εφαρμογής συλλογής δεδομένων Safe-R συλλέγονται σε μία κεντρική Βάση Δεδομένων και στη συνέχεια, αφού γίνει επεξεργασία τους, αποστέλλονται ως επεξεργασμένη πληροφορία πίσω στην εφαρμογή ώστε να αναπαρασταθούν στο χάρτη. Αυτή η διαδικασία απαιτεί τη λειτουργία μίας εφαρμογής, η οποία περιγράφηκε στο προηγούμενο κεφάλαιο, και ενός εξυπηρετητή (server) ο οποίος θα διαθέτει μία διεπαφή προγραμματισμού (API) και τη Βάση Δεδομένων. Στο παρόν κεφάλαιο θα περιγραφεί αναλυτικά ο εξυπηρετητής του συστήματος.

Η εφαρμογή του server ονομάστηκε **Safe-House** και η επιλογή της γλώσσας προγραμματισμού έγινε έτσι ώστε να δημιουργηθεί μία διεπαφή αρκετά απλή, γρήγορη και βολική στο χειρισμό απλών δομών δεδομένων, όπως αποτελούν τα δεδομένα που αποστέλλονται από την εφαρμογή Safe-R.

Επιλέχθηκε η γλώσσα προγραμματισμού **PHP** και συγκεκριμένα η υλοποίηση του **Safe-House** με χρήση του πλαισίου (framework) **Laravel**. Πέρα από την επιλογή της βασικής γλώσσας προγραμματισμού υπάρχουν αρκετές υποδομές, υποστηρικτικές του server.

Στις ενότητες που ακολουθούν γίνεται περιγραφή όλων των επιμέρους κομματιών που αποτελούν τα βασικότερα δομικά στοιχεία του **Safe-House**. Δίνονται οι λόγοι για τους οποίους έγινε η επιλογή της κάθε τεχνολογίας. Τέλος, περιγράφονται τα σημαντικότερα σημεία της υλοποίησης προγραμματιστικά, και παρουσίαση της βασικότερης λειτουργίας του Safe-House, δηλαδή της επεξεργασίας των δεδομένων.

4.1 Βασικές τεχνολογίες

Σε αυτή την ενότητα θα περιγραφούν κάποια βασικά δομικά στοιχεία του server και ο λόγος για τον οποίο επιλέχθηκε το κάθε ένα από αυτά. Βασικός γνώμονας για την επιλογή τους αποτέλεσε η επιλογή μόνο εφαρμογών ανοιχτού κώδικα (open source).

4.1.1 Το PHP Framework Laravel

Το PHP framework **Laravel** επιλέχθηκε ώστε να αποτελεί τη βασική υποδομή υλοποίηση της εφαρμογής **Safe-House**.

Το **Laravel** αποτελεί ένα, ανοιχτού κώδικα, διαδικτυακό πλαίσιο προγραμματισμού το οποίο δημιουργήθηκε από τον Taylor Otwell τον Ιούνιο του 2011. Παρά το μικρό χρονικό διάστημα στο οποίο το **Laravel** είναι διαθέσιμο στο ευρύ κοινό έχει καταφέρει να κερδίσει αρκετά την εμπιστοσύνη των προγραμματιστών και να έχει περισσότερα έργα κώδικα από ότι άλλα πολύ ωριμότερα framework, όπως το Symphony, το CakePHP, το CodeIgniter και το Yii (Bean, 2015).

Βασικός σκοπός του είναι η υλοποίηση διαδικτυακών εφαρμογών, οι οποίες ακολουθούν το μοντέλο **Model-View-Controller** (**MVC**). Το **Model–View–Controller** μοντέλο αποτελεί ένα μοντέλο αρχιτεκτονικής λογισμικού το οποίο χρησιμοποιείται για την δημιουργία περιβαλλόντων αλληλεπίδρασης χρήστη. Στο μοντέλο αυτό η εφαρμογή διαιρείται σε τρία διασυνδεδεμένα μέρη ώστε να διαχωριστεί η παρουσίαση της πληροφορίας στον χρήση από την μορφή που έχει αποθηκευτεί στο σύστημα. Το κύριο μέρος του μοντέλου είναι το αντικείμενο Model, το οποίο διαχειρίζεται την ανάκτηση / αποθήκευση των δεδομένων στο σύστημα. Το αντικείμενο View χρησιμοποιείται μόνο για να παρουσιάζεται η πληροφορία στον χρήστη. Το τρίτο μέρος είναι ο Controller ο οποίος δέχεται την είσοδο και στέλνει εντολές στο αντικείμενο Model και στο View (Coplien, 2009). Επομένως αποτελεί σε γενικές γραμμές ένα μοντέλο το οποίο ενισχύει το διαχωρισμό, προγραμματιστικά, διακριτών διαδικασιών μίας εφαρμογής με αποτέλεσμα την ευκολότερη υλοποίησή της, τη συντήρηση και την επεκτασιμότητα της.

Στο **Laravel** έχουν υλοποιηθεί συστήματα αυτοματισμού στη διαχείριση των Βάσεων Δεδομένων με χρήση μοντέλων, όπως επίσης και σύστημα αυθεντικοποίησης τύπου OAuth 2.0 και αρκετά ακόμη υποβοηθητικά συστήματα. Η έκδοση **5.1** του **Laravel**, η οποία και επιλέχθηκε, δίνει τη δυνατότητα χρονοπρογραμματισμού διεργασιών με επαναληπτικότητα, μία δυνατότητα η οποία υποστηρίζει και τη βασικότερη διεργασία του server, δηλαδή την επεξεργασία των δεδομένων.

4.1.2 Η Βάση Δεδομένων PostgreSQL

Βασικότερη προϋπόθεση επιλογής μίας Βάσης Δεδομένων ήταν η ικανότητά της να χειρίζεται χωρικά δεδομένα. Οι πιθανές επιλογές κατά την αναζήτηση και αξιολόγηση των διαφορετικών Βάσεων Δεδομένων ήταν οι εξής:

- **PostgreSQL**
- **IBM DB2**
- **Oracle**
- **Microsoft SQL Server**

Όλες οι προηγούμενες επιλογές δεν είναι ανοικτού κώδικα, πλην της πρώτης επιλογής (**PostgreSQL**). Όσων αφορά τα χωρικά δεδομένα και οι τέσσερις επιλογές προσφέρουν πληθώρα βοηθημάτων για χρήση τέτοιου είδους δεδομένων στα σχήματά τους. Μάλιστα, αρκετά από αυτά τα συστήματα βάσεων δεδομένων παρέχουν ειδικές επεκτάσεις αποκλειστικά για επεξεργασία χωρικών δεδομένων. Το πακέτο **PostGIS** για

την **PostgreSQL**, το **Oracle Spatial** για την **Oracle** και το **DB2 Spatial Extender** είναι μερικά παραδείγματα τέτοιων πακέτων.

Το πακέτο **PostGIS** παρέχει πολλές επιπρόσθετες δυνατότητες στη σχεσιακή Βάση Δεδομένων **PostgreSQL**. Συγκεκριμένα, προσθέτει ένα νέο τύπο δεδομένων, τη γεωμετρία (geometry), όπου μπορεί να χρησιμοποιηθεί ώστε να αναπαραστήσει σχήματα στο χάρτη (σημεία, γραμμές, πολύγωνα κλπ). Επίσης προστίθενται πολλές συναρτήσεις αυτοματισμού ώστε να υποστηρίζονται χωρικά ερωτήματα, όπως για παράδειγμα η εύρεση της απόστασης μεταξύ γεωμετριών, η εύρεση του εμβαδού κλπ. Τέλος, γίνεται χρήση ειδικών χωρικών ευρετηρίων (spatial indexes) τα οποία είναι προσαρμοσμένα ώστε να βελτιστοποιούν όλα τα SQL ερωτήματα που αφορούν γεωμετρίες (Columbia, 2005).

Το γεγονός της εύκολης εγκατάστασης σε Unix server, της παροχής ειδικού πακέτου υποστήριξης χωρικών δεδομένων και τέλος της εν γένει αποδοτικής λειτουργίας κατά το χειρισμό χωρικών δεδομένων ήταν οι βασικότεροι λόγοι επιλογής της **PostgreSQL** ως βάσης δεδομένων της εφαρμογής του server (Safe-House).

4.1.3 Ο εξυπηρετητής HTTP Nginx

Η επιλογή του εξυπηρετητή HTTP έγινε με γνώμονα την καλύτερη εξυπηρέτηση πολλαπλών παράλληλων συνδέσεων. Η χρήση της κλασσικότερης επιλογής εξυπηρετητή - του **Apache** - δεν ήταν η ενδεδειγμένη καθώς ο **Nginx** έχει αποδειχθεί αρκετά πιο αποδοτικός σε παράλληλες συνδέσεις (Nedelcu, 2010). Δεδομένου ότι τα επεξεργασμένα δεδομένα μπορεί να ζητηθούν από αρκετές συσκευές ταυτόχρονα η επιλογή ενός εξυπηρετητή HTTP ο οποίος έχει αντίστοιχες ικανότητες αποτελεί μονόδρομο. Επομένως ο **Nginx** αποτέλεσε την τελική επιλογή.

4.2 Το Σχήμα της Βάσης Δεδομένων

Δύο από τους βασικότερους πίνακες της Βάσης Δεδομένων αφορούν τα δύο κεντρικά μοντέλα Frame και Snapshot. Επίσης υπάρχουν αρκετοί πίνακες οι οποίοι λειτουργούν βοηθητικά για την αυθεντικοποίηση των χρηστών μέσω του OAuth 2.0 (π.χ. πίνακες oauth_access_tokens, oauth_grants κλπ) καθώς επίσης και για την αποθήκευση των ίδιων των χρηστών (πίνακας Users).

Ο πλέον σημαντικότερος πίνακας όμως είναι ο πίνακας ο οποίος περιέχει τους δρόμους της περιοχής δοκιμών και ονομάζεται **"roads"**. Εν αντιθέσει με τους υπόλοιπους πίνακες ο πίνακας **roads** είναι ήδη προκατασκευασμένος από τον οργανισμό OpenStreetMap και διατίθεται μαζί με αντίστοιχους χάρτες για όλο τον κόσμο από το διαδικτυακό τόπο download.geofabrik.de.

Το αρχείο το οποίο είναι διαθέσιμο είναι της μορφής .pbf. Η κατάληξη .pbf είναι μία εναλλακτική κατάληξη για XML αρχεία τα οποία αφορούν γεωχωρικά δεδομένα. Σχεδιάστηκε με βασικό γνώμονα την επεκτασιμότητα, την ταχύτητα ανάγνωσης / εγγραφής και τέλος τη δυνατότητα συμπίεσης.

Καθώς, όμως, απαιτείται να έχουμε τα δεδομένα των δρόμων σε μορφή διαχειρίσιμη από τη PostgreSQL ΒΔ έπρεπε να γίνει μετατροπή του .pbf αρχείου σε SQL εντολές και στη συνέχεια εισαγωγή των δεδομένων στον πίνακα **roads**. Αυτή η διαδικασία έγινε με χρήση του εργαλείου **osm2postgresql**. Ένα εργαλείο ανοιχτού κώδικα διαθέσιμου μέσω της ιστοσελίδας sourceforge.net[1]. Αποτέλεσμα της επεξεργασίας του αρχείου .pbf του χάρτη της Ελλάδος είναι ένα αρχείο τύπου SQL το οποίο δημιουργεί και συμπληρώνει έναν πίνακα με όλους τους δρόμους της Ελλάδας όπως αυτοί είναι καταχωρημένοι από τους χρήστες του οργανισμού OpenStreetMap.

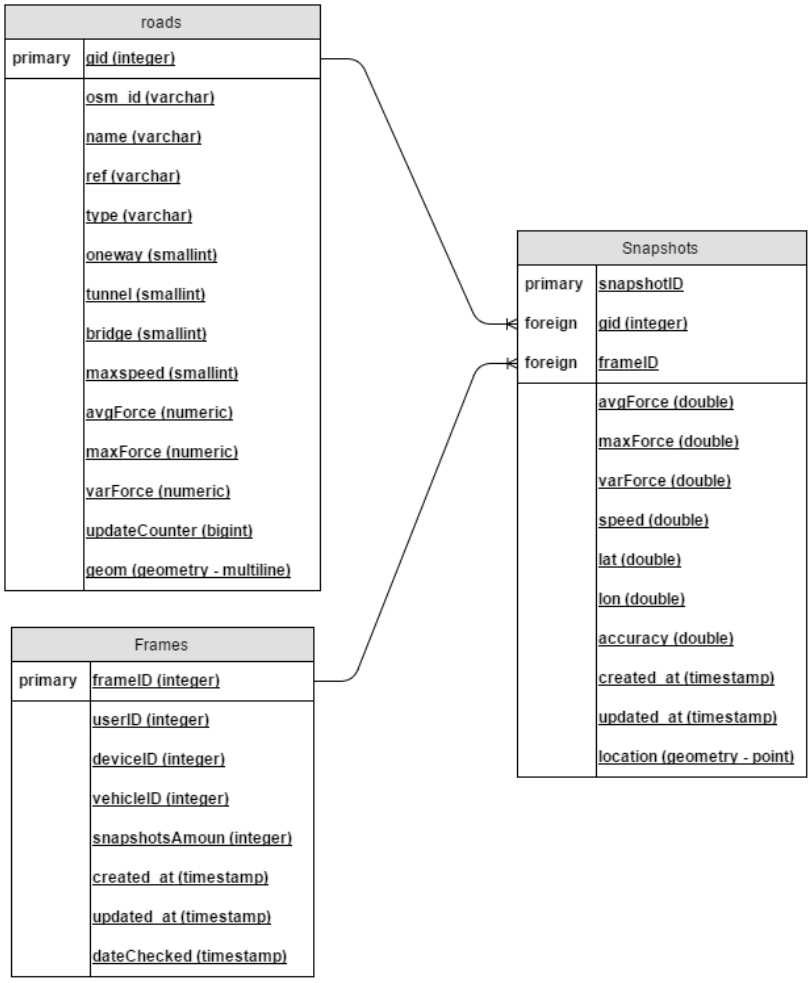
Για λόγους καλύτερης απόδοσης έπρεπε να αφαιρεθούν, χειροκίνητα, όσοι δρόμοι δεν ανήκουν στην περιοχή δοκιμών και όσοι δρόμοι εντός της περιοχής δοκιμών δεν αντιστοιχούν σε προσβάσιμους με αυτοκίνητο δρόμους. Έτσι, εκτός από όλους τους εκτός περιοχής δοκιμών δρόμους, αφαιρέθηκαν όλα τα μονοπάτια, οι πεζόδρομοι, η πεζογέφυρες, οι παράδρομοι, οι ποδηλατοδρόμοι και, γενικότερα, όλοι οι μη δημόσιοι αυτοκινητόδρομοι. Το τελικό αποτέλεσμα είναι **ένας πίνακας με 2234 δρόμους**.

Επίσης, στον πίνακα **roads** έπρεπε να δημιουργηθούν και πέντε στήλες οι οποίες αφορούν καθαρά την εφαρμογή Safe-R και έχουν να κάνουν με τα δεδομένα που συλλέγονται σε κάθε Snapshot. Έτσι δημιουργήθηκαν οι εξής στήλες:

- **avgForce** - Αφορά τη μέση τιμή των avgForce (μέσης επιτάχυνσης) τιμών κάθε Snapshot.
- **varForce** - Αφορά τη μέση τιμή των varForce (διασπορά επιτάχυνσης) τιμών κάθε Snapshot.
- **maxForce** - Αφορά τη μέση τιμή των maxForce (μέγιστης επιτάχυνσης) τιμών κάθε Snapshot.
- **avgSpeed** - Αφορά τη μέση τιμή της ταχύτητας κάθε Snapshot.
- **updateCounter** - Μετρητής του συνόλου των Snapshot που έχουν καταχωρηθεί στο συγκεκριμένο δρόμο.

[1] Πηγή: https://sourceforge.net/projects/osm2postgresql/

Στην Εικόνα 10 υπάρχει ένα σχεδιάγραμμα με τους τρεις βασικότερους πίνακες, όπως αυτοί ορίστηκαν προηγουμένως.



Εικόνα 10 - Σχεδιάγραμμα ΒΔ

Μπορούμε να παρατηρήσουμε το συσχετισμό των πινάκων με χρήση των κλειδιών **gid** (πίνακες **roads** - **Snapshots**) και **frameID** (πίνακες **Frames** - **Snapshots**).

Τέλος, αξίζει να αναφερθεί πως για λόγους απόδοσης έχει δημιουργηθεί ευρετήριο (index), εκτός για τα κλειδιά του κάθε πίνακα, και για τη στήλη γεωμετριών (**geom)** του πίνακα **roads**. Το ευρετήριο αυτό αποτελεί ένα ειδικού τύπου ευρετήριο, το οποίο ονομάζεται γεωχωρικό ευρετήριο (spatial index). Τέτοιου είδους γεωχωρικά ευρετήρια χρησιμοποιούνται ώστε να επιταχύνονται σημαντικά ερωτήματα SQL που αφορούν γεωχωρικά δεδομένα. Παρακάτω υπάρχει η SQL εντολή η οποία μας επιτρέπει να δημιουργήσουμε το συγκεκριμένο spatial index.

```
CREATE INDEX roads_geom_idx
  ON public.roads
  USING gist
  (geom);
```

4.3 Λήψη δεδομένων

Όπως προαναφέρθηκε, η λογική με την οποία έχει υλοποιηθεί ο server ακολουθεί το μοντέλο MVC. Σύμφωνα με το MVC μοντέλο κάθε διεπαφή αντιστοιχεί σε μία μέθοδο (**action**) και όλες οι σχετικές μέθοδοι-διεπαφές που σχετίζονται με ένα αντικείμενο ομαδοποιούνται σε μία κλάση που ονομάζεται **Controller**.

Για να πραγματοποιηθεί η λήψη δεδομένων σε μορφή JSON, ο server παρέχει μία αντίστοιχη διεπαφή. Η μέθοδος η οποία έχει αναλάβει τη λήψη ονομάζεται **store** και η κλάση τύπου **Controller** η οποία εμπεριέχει αυτή τη μέθοδο είναι η κλάση **FramesController**.

```
public function store() {
   try {
      $data = json_decode(Input::get('data'), true);
      $frame = $data['frame'];
      $snapshots = $data['snapshots'];

      if (count($snapshots) != $frame['snapshotsAmount']) {
         return $this->respondErrorWithDescriptionAndCode('Frame\'s snapshotsAmount do not match
count of Snapshots.');
      }

      $frameCreated = Frame::create($frame);
      foreach ($snapshots as $snapshot) {
         try {
            $snapshot['frameID'] = $frameCreated->frameID;
            $model = new Snapshot($snapshot);
            $model->save();
         } catch (\ErrorException $e) {
            throw new Exception("Error on Snapshot model creation.");
         }
      }
   } catch (Exception $e) {
      return $this->setStatusCode(Response::HTTP_INTERNAL_SERVER_ERROR)-
>respondErrorWithDescriptionAndCode('An error occurred during data storing in DB.');
   }
   return $this->respondCreated('Frame & Snapshots added in database.');
}
```

Αρχικά, η μέθοδος **store** αποκωδικοποιεί το απεσταλμένο JSON (json_decode μέθοδος). Στη συνέχεια, και εφόσον έχει γίνει επιτυχώς η αποκωδικοποίηση, πραγματοποιείται μία τυπική καταμέτρηση των απεσταλμένων **Snapshots** έτσι ώστε να γίνει σύγκριση με το δείκτη **snapshotsAmount** που εμπεριέχεται στα δεδομένα. Σε περίπτωση μη ταυτοποίησης η διαδικασία σταματάει. Παρακάτω υπάρχει ένα δείγμα του αρχείου JSON που καλείται να αποκωδικοποιήσει η μέθοδος.

```
{
  "data": {
    "frame": {
      "userID": 1,
      "deviceID": 1,
      "vehicleID": 1,
      "snapshotsAmount": 2
    },
    "snapshots": [
      {
        "avgForce": "11.2314",
        "maxForce": "12.8673",
        "varForce": "5.634",
        "speed": "45.2",
        "lat": "21.12312411",
        "lon": "38.11251213",
        "accuracy": "4"
      },
      {
        "avgForce": "9.924",
        "maxForce": "10.343",
        "varForce": "2.132",
        "speed": "44.4",
```

```
        "lat": "21.1232521",
        "lon": "38.113421",
        "accuracy": "4"
      }
    ]
  }
}
```


Ακολουθεί η δημιουργία ενός μοντέλου τύπου **Frame** και η αποθήκευσή του στη ΒΔ. Έπειτα από το βασικό μοντέλο **Frame** δημιουργούνται σειριακά όλα τα μοντέλα για τα **Snapshots** και αποθηκεύονται στον αντίστοιχο πίνακα **Snapshots** στη ΒΔ. Έτσι αποθηκεύονται τα ανεπεξέργαστα δεδομένα στη ΒΔ ώστε στη συνέχεια να πραγματοποιηθεί επεξεργασία τους και εξαγωγή πληροφορίας από αυτά.

Μετά το πέρας της διαδικασίας λήψης ο server αποστέλλει μία απάντηση η οποία αποτελεί ένα ακόμη αρχείο τύπου JSON. Σε περίπτωση όπου όλη η διαδικασία έχει ολοκληρωθεί με επιτυχία ο κωδικός HTTP (HTTP response code) είναι **200**. Έτσι ενημερώνεται η εφαρμογή για την επιτυχή αποστολή και αποθήκευσή των δεδομένων στο server. Σε περίπτωση οποιουδήποτε προβλήματος υπάρχει αντίστοιχη απάντηση αλλά με HTTP response code **500** υποδηλώνοντας πρόβλημα από τη μεριά του server είτε με κωδικό **400** υποδηλώνοντας λάθος στο απεσταλμένο αρχείο JSON αντίστοιχα.

Η απάντηση σε περίπτωση επιτυχούς εισαγωγής των δεδομένων στη Βάση Δεδομένων είναι η εξής:


```
{
  "data": "Frame & Snapshots added in database.",
  "error": {
    "description": "",
    "code": ""
  }
}
```


Και σε περίπτωση λάθους κατά την εισαγωγή δεδομένων υπάρχει μία απάντηση όπως η παρακάτω.


```
{
  "data": null,
  "error": {
    "description": "Unexpected error. Try again later.",
    "code": 500
  }
}
```

4.4 Επεξεργασία δεδομένων

Όπως έχει προαναφερθεί τα βασικά δεδομένα που διατηρούνται στη Βάση Δεδομένων είναι τα **Frames** και τα **Snapshots**. Ένα **Frame** ομαδοποιεί και εκπροσωπεί ένα σύνολο **Snapshots**. Κάθε **Snapshot** περιέχει δεδομένα αναταραχής για ένα συγκεκριμένο γεωχωρικό σημείο.

Η βασική επεξεργασία που επιδέχονται αυτά τα δεδομένα είναι η αντιστοίχισή κάθε **Snapshot** σε ένα δρόμο από τον πίνακα δρόμων (roads) με βάση την απόσταση του **Snapshot** από αυτόν. Επίσης γίνεται εκκαθάριση των **Snapshot** που μπορεί να θεωρηθούν μη αξιόπιστα με βάση την ταχύτητα, την ακρίβεια του GPS αλλά και τη λογική της τήρησης ενός μονοπατιού στο σύνολο μίας ομάδας **Snapshot**.

Η διαδικασία της επεξεργασία των δεδομένων ακολουθεί μία λογική επαναλαμβανόμενης διαδικασίας. Ο server πρέπει ανά πάσα στιγμή να είναι σε θέση να επεξεργαστεί δεδομένα που αποστέλλονται σε αυτόν οπότε υπάρχει μία χρονοπρογραμματισμένη διεργασία η οποία ανά ένα λεπτό ελέγχει τη Βάση Δεδομένων και εάν εντοπίσει δεδομένα τα οποία δεν έχουν υποστεί επεξεργασία τα επιλέγει προς επεξεργασία.

Η μέθοδος υπεύθυνη για την επεξεργασία βρίσκεται στην κλάση **UpdateRoadsTable**, η οποία περιγράφεται στην επόμενη ενότητα.

4.4.1 Η κλάση UpdateRoadsTable

Η κλάση **UpdateRoadsTable** αποτελεί μία κλάση τύπου **Command**. Οι κλάσεις τύπου **Command**, στο Laravel Framework, αποτελούν κλάσεις μικρών αυτόνομων εκτελέσιμων αρχείων που έχουν τη δυνατότητα να εκτελεστούν είτε μέσω της γραμμής εντολών είτε αυτόματα μέσω του Framework σε συνεργασία με προγράμματα χρονοπρογραμματισμού. Ένα κλασσικό παράδειγμα προγράμματος χρονοπρογραμματισμού στα Unix αποτελεί το **Crontab**.

Αρχικά έχουν οριστεί οι μεταβλητές **name** και **description**. Οι δύο αυτές μεταβλητές, είναι προκαθορισμένες από το Laravel και αποτελούν το όνομα με το οποίο καλείται η μέθοδος από τη γραμμή εντολών και την περιγραφή της λειτουργίας της μεθόδου αντίστοιχα. Οι τιμές που δόθηκαν στις δύο αυτές μεταβλητές είναι οι ακόλουθες:

- **name** - 'roads:update'
- **description** - 'Update roads table with not already imported Snapshots data.'

Από το όνομα και την περιγραφή μπορεί να γίνει εύκολα αντιληπτό ότι η κλάση **UpdateRoadsTable**, όταν εκτελείται, ανανεώνει τον πίνακα **roads** με δεδομένα τύπου **Snapshot** που δεν έχουν ακόμη υποστεί επεξεργασία.

Στην ίδια κλάση υπάρχει και η μέθοδος **fire**. Η συγκεκριμένη μέθοδος είναι αυτή που καλείται κατά την εκτέλεση της διαδικασίας επεξεργασίας των δεδομένων. Παρακάτω υπάρχει μέρος του κώδικα της μεθόδου **fire**.

```
$uncheckedFrames = Frame::selectUnchecked();
foreach ($uncheckedFrames as $frame) {
   $frame->updateRoads();
}
```

Αυτές οι τρεις γραμμές κώδικα αρχικά βρίσκουν όσα **Frame** υπάρχουν στη Βάση Δεδομένων, και τα **Snapshot** τα οποία περιέχουν δεν έχουν υποστεί επεξεργασία, με την κλήση της μεθόδου **selectUnchecked** και στη συνέχεια ένα-ένα αυτά τα **Frame** εκτελούν την μέθοδο **updateRoads**. Στην επόμενη ενότητα ακολουθεί περεταίρω ανάλυση των μεθόδων **selectUnchecked** και **updateRoads** ώστε να γίνει πιο

κατανοητή η διαδικασία επεξεργασίας των δεδομένων και ανανέωσης του πίνακα των δρόμων από το σύστημα.

4.4.2 Ανανέωση του πίνακα δρόμων

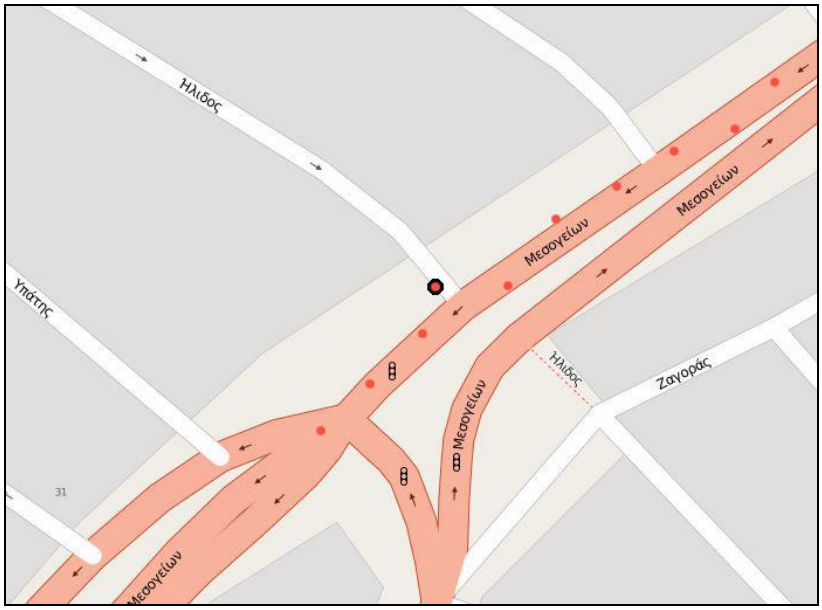
Συνεχίζοντας από την προηγούμενη ενότητα, η μέθοδος **selectUnchecked** επιλέγει από τον πίνακα **Frames** όσα **Frame** έχουν κενή ημερομηνία ελέγχου (στήλη **dateChecked**) και τα επιστρέφει. Κάθε φορά που ένα Frame υπόκειται σε επεξεργασία αυτή η στήλη γεμίζει με την ημερομηνία ελέγχου. Ακολουθεί ο κώδικας της μεθόδου.

```
public static function selectUnchecked() {
   return self::whereNull('dateChecked')->orderBy('frameID')->get();
}
```

Η μέθοδος **updateRoads** αναλαμβάνει να επιλέξει τα χρήσιμα μόνο **Snapshot** του εκάστοτε **Frame** και στη συνέχεια να καλέσει τη μέθοδο **updateClosestRoad** για κάθε ένα **Snapshot** και τέλος να ορίσει την ημερομηνία ελέγχου στο συγκεκριμένο **Frame**. Έτσι σε επόμενη κλήση της κλάσης **UpdateRoadsTable** δε θα επιλεγεί ξανά το συγκεκριμένο **Frame**.

```
function updateRoads() {
   foreach ($this->_selectSnapshotsAvoidingIntersections() as $snapshot) {
      $snapshot->updateClosestRoad();
   }
   $this->_updateDateChecked();
}
```

Κατά τη διαδικασία συλλογής δεδομένων είναι λογικό, παρά τους περιορισμούς σε ακρίβεια του GPS που έχουμε ορίσει στη συσκευή συλλογής, να υπάρχουν **Snapshot** τα οποία να αποκλίνουν μερικώς από το μονοπάτι δειγματοληψίας. Έτσι όταν το όχημα κινείται σε ένα δρόμο υπάρχει η πιθανότητα κάποια **Snapshot** να θεωρηθούν λανθασμένα, λόγω της θέσης τους, ότι ανήκουν σε κάποιο κάθετο παράδρομο παρά στον βασικό δρόμο που δειγματοληπτείται. Είναι πιο εύκολα αντιληπτό αυτό το φαινόμενο παρατηρώντας μία εικόνα με ένα παράδειγμα δειγματοληψίας.



Εικόνα 11 - Παράδειγμα Δειγματοληψίας

Στην Εικόνα 11 παρατηρούμε ένα στιγμιότυπο δειγματοληψίας από ένα όχημα κινούμενο στη λεωφόρο Μεσογείων καθώς αυτό προσπερνάει την οδό Ήλιδος. Κάθε κόκκινο σημάδι στο χάρτη αποτελεί ένα στιγμιότυπο καταγραφής, δηλαδή ένα **Snapshot**. Παρατηρούμε ότι ενώ τα περισσότερα **Snapshot** που έχει δημιουργήσει η συσκευή καταγραφής βρίσκονται επάνω στη λεωφόρο, υπάρχει και ένα το οποίο αν και

θα έπρεπε να συσχετισθεί με τη λεωφόρο Μεσογείων φαίνεται να αντιστοιχεί στην οδό Ήλιδος. Είναι το **Snapshot** το οποίο φαίνεται με το πιο έντονο περίγραμμα στην εικόνα.

Οπότε, υπάρχει η ανάγκη εξάλειψης αυτού του είδους των δεδομένων. Ακριβώς για αυτό το λόγο υπάρχει η μέθοδος **_selectSnapshotsAvoidingIntersections**. Η συγκεκριμένη μέθοδος, η οποία φαίνεται παρακάτω, είναι η βασικότερη μέθοδος κατά τη διαδικασία ανανέωσης του πίνακα δρόμων.

```
private function _selectSnapshotsAvoidingIntersections() {
   $snapshotIdsForFiltering = $this->_getSnapshotIdsForRejection($this-
>_getClusteredRoadsFromSnapshots());
   return $this->selectValuableData($snapshotIdsForFiltering);
}
```

Η μέθοδος **_selectSnapshotsAvoidingIntersections** πρέπει να επιστρέψει στη μέθοδο **updateRoads** μία λίστα με όλα τα **Snapshot** έχοντας αναθέσει σε αυτά ένα συγκεκριμένο δρόμο (τον κοντινότερο σε κάθε ένα). Μάλιστα, θα πρέπει να έχουν επιλεγεί μονάχα τα χρήσιμα **Snapshot** και όχι όλα όσα ανήκουν σε ένα **Frame**.

Η μέθοδος **_getSnapshotIdsForRejection** καλείται να κάνει έναν αρχικό έλεγχο όλων των **Snapshot** στο συγκεκριμένο **Frame** και να εντοπίσει όσα περισσότερα **Snapshot** βρίσκονται, λόγω απόκλισης του GPS, πιο κοντά σε λάθος δρόμο. Η λίστα με αυτούς τους δρόμους δίνεται ως είσοδος στη μέθοδο **selectValuableData** έτσι ώστε να μη συμπεριληφθούν εκείνα τα δεδομένα στο τελικό σύνολο των **Snapshot** που θα χρησιμοποιηθούν. Παράλληλα με τον έλεγχο εντοπισμού λανθασμένων **Snapshot** γίνεται και μία πολύ σημαντική διαδικασία για τη συνέχεια, η ανάθεση δρόμου σε κάθε **Snapshot**.

Η διαδικασία ανάθεσης αρχίζει με την εύρεση του πλησιέστερου δρόμου από τον πίνακα με τις γεωμετρίες των δρόμων (πίνακας roads). Κάθε **Snapshot** περιέχει το γεωγραφικό μήκος και πλάτος της δειγματοληψίας (μεταβλητές **lon** και **lat**). Ο ακόλουθος κώδικας, της μεθόδου **findNearestRoad**, είναι σε θέση να εντοπίσει την πλησιέστερη γεωμετρία δρόμου αναλύοντας τις αποστάσεις όλων των διαθέσιμων δρόμων στη Βάση Δεδομένων (**ST_Distance**) και στη συνέχεια κάνοντας μία αύξουσα ταξινόμηση των αποτελεσμάτων (**ORDER BY** distance **ASC**) και επιλέγοντας απλά την πρώτη γεωμετρία-δρόμο από τα αποτελέσματα (**LIMIT 1**).

```
public function findNearestRoad() {
   return DB::select('
     SELECT gid,
            ST_Distance(
               ST_SetSRID(ST_Point(' . $this->lon . ', ' . $this->lat . '),4326), geom
            ) as distance,
            name,
            geom
     FROM roads
     ORDER BY distance ASC
     LIMIT 1')[0];
}
```

Στη συνέχεια, και εφόσον έχουμε εντοπίσει τον πιο κοντινό δρόμο στο εκάστοτε **Snapshot**, γίνεται μία ανάθεση του κλειδιού του δρόμου (**gid**) στην αντίστοιχη στήλη **gid** του πίνακα **Snapshot**. Έτσι κάθε **Snapshot** έχει γνώση του δρόμου στον οποίο πλέον ανήκει.

Τέλος, η μέθοδος **selectValuableData** θα επιστρέψει όλα τα **Snapshot** εκείνα τα οποία δεν ανήκουν στη λίστα με τα **Snapshot** προς αποφυγή έχουν συγκεκριμένα όρια σε ταχύτητα και ακρίβεια GPS. Ακολουθεί ο σχετικός κώδικας.

```
function selectValuableData($idForRejection = []) {
   return Snapshot::where('speed', '>=', self::SPEED_LOW_THRESHOLD)
      ->where('speed', '<', self::SPEED_HIGH_THRESHOLD)
      ->where('accuracy', '<=', self::ACCURACY_THRESHOLD)
      ->where('frameID', '=', $this->frameID)
      ->whereNotIn('snapshotID', $idForRejection)
      ->orderBy('snapshotID')
      ->get();
}
```

Επανερχόμενοι στη μέθοδο **updateRoads**, κάθε **Snapshot** το οποίο επιστράφηκε από τη μέθοδο _**selectSnapshotsAvoidingIntersections** θα εκτελέσει τη μέθοδο **updateClosestRoad**. Η μέθοδος **updateClosestRoad**, η οποία ανήκει στην κλάση **Snapshot**, αναλαμβάνει να ανανεώσει τις εξής στήλες του κάθε δρόμου:

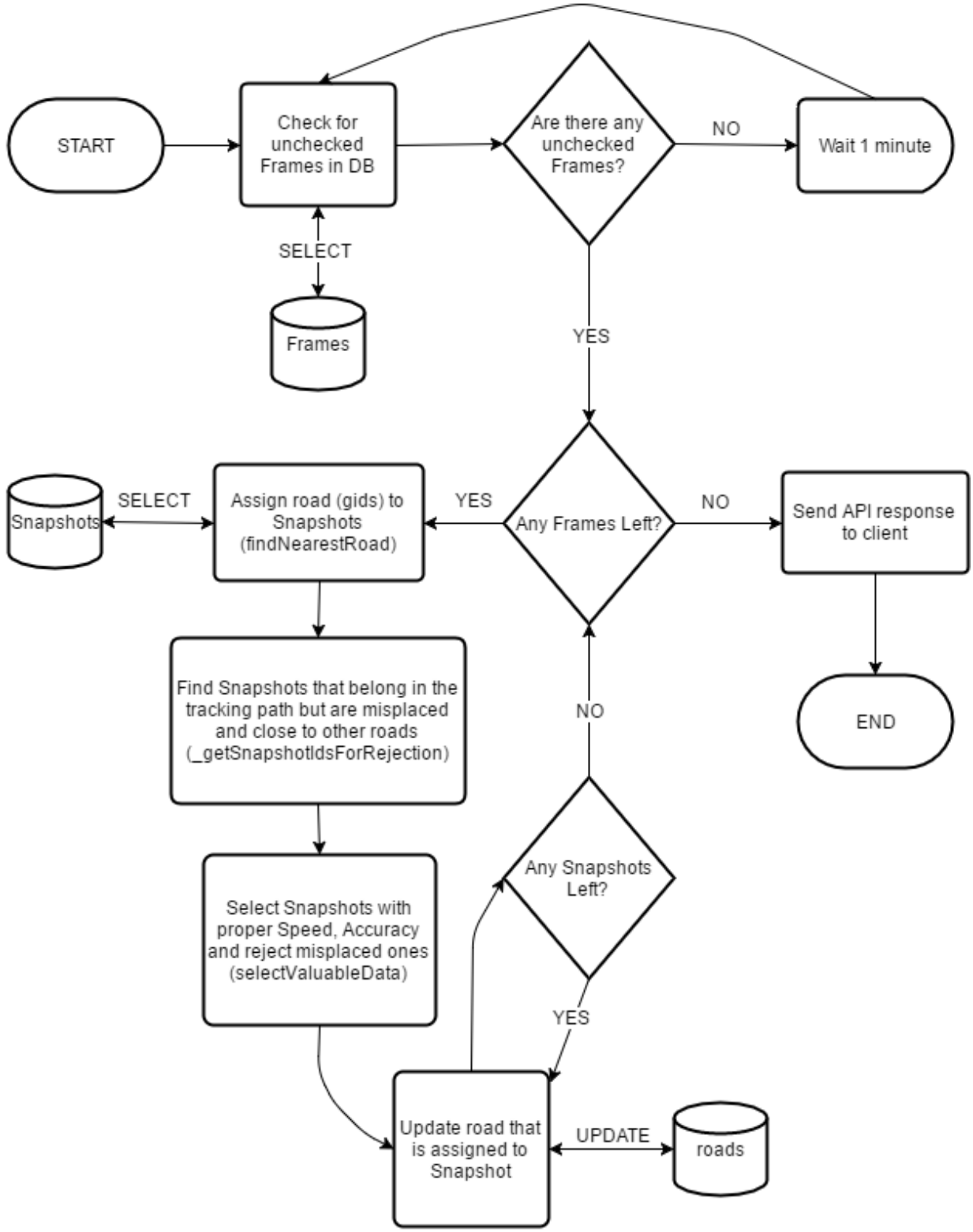
- **avgForce**
- **maxForce**
- **varForce**
- **avgSpeed**
- **updateCounter**

Έτσι, και μετά την ανανέωση των δρόμων στη ΒΔ έχει ολοκληρωθεί ο κύκλος επεξεργασίας των δεδομένων. Αυτή η διαδικασία εκτελείται κάθε λεπτό στον server και κάθε φορά εντοπίζονται τα Frame εκείνα τα οποία δεν έχουν υποστεί επεξεργασία.

Η μέθοδος η οποία ορίζει το κάθε πότε πρέπει να εκτελείται η συγκεκριμένη κλάση ονομάζεται **schedule** και ανήκει στην κλάση Kernel. Ο κώδικάς της, που βρίσκεται παρακάτω, κάνει σαφές ότι πρέπει να εκτελεστεί η διαδικασία ανανέωσης του πίνακα των δρόμων κάθε λεπτό (**everyMinute**).

```
protected function schedule(Schedule $schedule)
{
   $schedule->command('roads:update')
      ->everyMinute()
      ->appendOutputTo(storage path('logs/schedule.log'));
}
```

Για να γίνει περισσότερο αντιληπτή η αλληλουχία των ενεργειών κατά τη διαδικασία ανανέωσης του πίνακα των δρόμων, ακολουθεί ένα σχεδιάγραμμα που περιγράφει όλη τη διαδικασία. Στην Εικόνα 12 φαίνεται διαγραμματικά η διαδικασία ανανέωσης του πίνακα των δρόμων όπως επίσης και οι προσβάσεις της Βάσης Δεδομένων από την εφαρμογή του server.



Εικόνα 12 - Η Διαδικασία Ανανέωσης του Πίνακα Δρόμων

4.5 Αποστολή δεδομένων

Η τρίτη και τελευταία βασική λειτουργία του server αφορά την αποστολή των δεδομένων που αφορούν τους δρόμους και τις αναταραχές που έχουν καταγραφεί σε αυτούς κατά την επεξεργασία των δεδομένων.

Η κλάση που είναι υπεύθυνη ώστε να διαχειρίζεται αιτήματα αναζήτησης δρόμων είναι η κλάση **RoadsController** και η μέθοδος (action) η οποία καλείται όταν υπάρχει ένα τέτοιου είδους αίτημα είναι η **index**.

Η μέθοδος **index,** της οποίας ο κώδικας βρίσκεται παρακάτω, δέχεται ως ορίσματα τέσσερις παραμέτρους που αφορούν δύο γεωχωρικά σημεία. Τα δύο αυτά σημεία προέρχονται από το ορατό μέρος του χάρτη στην εφαρμογή Safe-R. Πιο συγκεκριμένα, αφορούν το βορειοδυτικό και το νοτιοανατολικό άκρο του χάρτη. Έτσι, ως **left** και **top** ορίζονται το γεωγραφικό μήκος και πλάτος του βορειοανατολικού άκρου αντίστοιχα και ως **right** και **bottom** το μήκος και πλάτος του νοτιοανατολικού άκρου.

```
public function index($left, $bottom, $right, $top) {
   $roads = Road::findRoadsIntersectingBoundingBox($left, $bottom, $right, $top);
   return $this->respond([
      'data' => $this->roadTransformer->transformCollection($roads),
      'error' => [
         'description' => '',
         'code' => ''
      ]
   ]);
}
```

Ο τρόπος με τον οποίο μπορεί να καλεστεί η μέθοδος **index** είναι με μία απλή πρόσβαση σε ένα σύνδεσμο με την παρακάτω μορφή:

http://www.safe-r.gr/api/v1/roads/left/bottom/right/top

Η μέθοδος **index** καλεί τη μέθοδο **findRoadsIntersectingBoundingBox** της κλάσης **Road**. Ως παράμετροι για τη μέθοδο αυτή δίνονται οι τέσσερις παράμετροι (left, bottom, right, top).

Στη μέθοδο **findRoadsIntersectingBoundingBox** γίνεται ένα ερώτημα SQL στη ΒΔ της παρακάτω μορφής:

```
public static function findRoadsIntersectingBoundingBox($left, $bottom, $right, $top) {
   return DB::select('
     SELECT gid, "avgForce", "maxForce", "varForce", "avgSpeed", "updateCounter",
            ST_AsGeoJSON(ST_Transform(geom, 4326), 7)
     FROM roads
     WHERE geom && ST_MakeEnvelope('.$left.','.$bottom.','.$right.','.$top.',4326) AND
           "updateCounter" > 0');
}
```

Το αποτέλεσμα της μεθόδου είναι μια λίστα με δρόμους οι οποίοι ανήκουν στο τετράγωνο που ορίζεται από τα δοθέντα γεωχωρικά σημεία. Επιστρέφονται τα στοιχεία που αφορούν τα εξής πεδία στον πίνακα roads:

- avgForce
- maxForce
- varForce
- avgSpeed
- updateCounter
- geom

Το τελευταίο πεδίο αφορά τη γεωμετρία του δρόμου. Είναι όλα τα σημεία από τα οποία αποτελείται ο δρόμος σε μορφή JSON.

Ένα παράδειγμα της JSON απάντησης που στέλνει ο server παρατίθεται παρακάτω:


```
{
  "data": [
    {
      "gid": 32965,
      "averageForce": 9.85,
      "maxForce": 11.21,
      "varForce": 0.62,
      "averageSpeed": 27.96,
      "updateCounter": 2,
      "points": [
        [
          23.7669079,
          37.9852547
        ],
        [
          23.7668716,
          37.9855327
        ],
        [
          23.7667514,
          37.9856409
        ],
        [
          23.7665359,
          37.9858007
        ],
        [
          23.7661959,
          37.9860422
        ],
        [
          23.7658047,
          37.9863535
        ],
        [
          23.765708,
          37.9864305
        ]
      ]
    },
    {
      "gid": 60348,
      "averageForce": 9.8,
      "maxForce": 10.85,
      "varForce": 0.83,
      "averageSpeed": 29.75,
      "updateCounter": 2,
      "points": [
        [
          23.7666587,
          37.9854405
        ],
        [
          23.7667771,
          37.9854681
        ],
        [
          23.7668716,
          37.9855327
        ]
      ]
    }
  ],
  "error": {
    "description": "",
    "code": ""
  }
}
```


Στο παράδειγμα αυτό ο server επιστρέφει ένα JSON αρχείο με δύο μόνο δρόμους. Ο πρώτος αποτελείται από εφτά σημεία και ο δεύτερος από τρία. Αυτές οι λίστες με τα σημεία του κάθε δρόμου θα αποτελέσουν, αργότερα, τις οδηγίες με τις οποίες θα μπορέσει η εφαρμογή Safe-R να σχεδιάσει το δρόμο επάνω στο χάρτη με το κατάλληλο χρώμα. Το κατάλληλο χρώμα θα επιλεγεί από την εφαρμογή με βάση τα επιπρόσθετα

στοιχεία που παρέχονται μαζί σε κάθε εγγραφή δρόμου που επιστρέφεται. Αυτά αφορούν τη μέση, μέγιστη και διασπορά τιμών για τα μέτρα των επιταχύνσεων που έχουν καταγραφεί καθώς επίσης και η μέση ταχύτητα του οχήματος καταγραφής.

5. Πειράματα - Συμπεράσματα

Στο παρόν κεφάλαιο θα γίνει μία παρουσίαση των αποτελεσμάτων της καταγραφής όπως αυτά εμφανίζονται στον ηλεκτρονικό χάρτη της εφαρμογής Safe-R. Επίσης, θα γίνει μία σύγκριση των αποτελεσμάτων αυτών με την πραγματική εικόνα των δρόμων αυτών, η οποία έχει καταγραφεί με χρήση ψηφιακής φωτογραφικής μηχανής.

Θα γίνει μελέτη τριών δρόμων, οι οποίοι αποτελούν παράδειγμα και για τις τρεις πιθανές εκφάνσεις ποιότητας δρόμου, όπως αυτές έχουν οριστεί κατά τη διαδικασία της δημιουργίας της εφαρμογής.

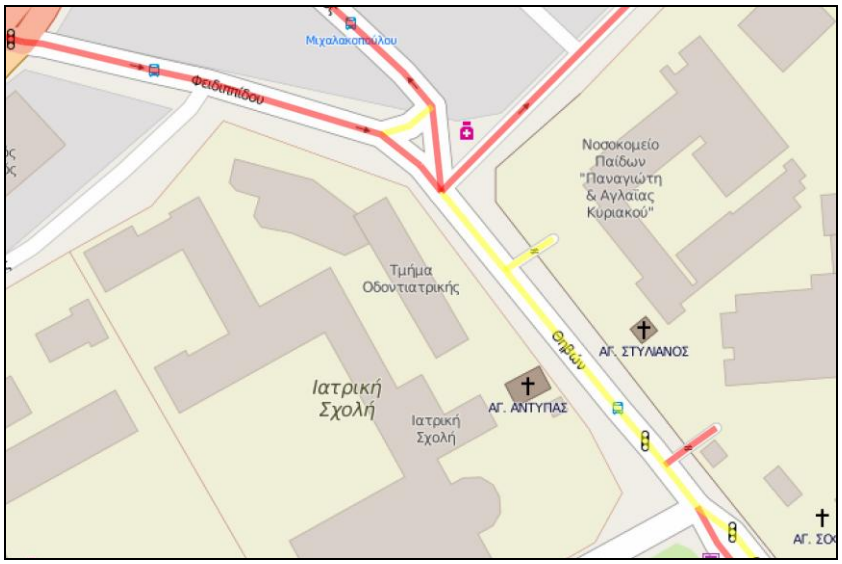
Οι δρόμοι που επιλέχθηκαν ανήκουν στην περιοχή των Αμπελοκήπων (δήμος Αθηναίων) και του Γουδί (δήμος Ζωγράφου). Τα οδικά τμήματα που επιλέχθηκαν ανήκουν στις οδούς:

- Φειδιππίδου (τμήμα από λεωφόρο Μεσογείων έως Θηβών)
- Θηβών
- Σπηλιωτοπούλου (τμήμα από Θηβών έως Δίκης)

Όπως έχει αναφερθεί και σε προηγούμενο κεφάλαιο οι τρεις πιθανές καταστάσεις ενός δρόμου είναι οι εξής:

- Καλή (μπλε χρώμα)
- Μέτρια (κίτρινο χρώμα)
- Κακή (κόκκινο χρώμα)

Στην Εικόνα 13 μπορούμε να παρατηρήσουμε τα δύο από αυτά τα χρώματα. Για την ακρίβεια στο παράδειγμα αυτό βλέπουμε την καταγραφή των αναταραχών από τη συσκευή για τα τμήματα των οδών Φειδιππίδου και Θηβών (έως την οδό Μικράς Ασίας).



Εικόνα 13 - Οδοί Φειδιππίδου & Θηβών

Η εφαρμογή Safe-R παρουσιάζει ως κακή ποιότητα οδοστρώματος την οδό Φειδιππίδου (κόκκινο χρώμα) και ως μέτρια το τμήμα της Θηβών (κίτρινο χρώμα), το οποίο για χάριν ευκολίας θα αποκαλούμε ως πρώτο τμήμα από εδώ και στο εξής. Παρακάτω γίνεται ανάλυση του κάθε τμήματος χωριστά.

5.1 Οδός Φειδιππίδου (κακή ποιότητα)

Στην Εικόνα 14 - Φειδιππίδου (Πανοραμική)Εικόνα 14 βλέπουμε μία πανοραμική εικόνα της οδού Φειδιππίδου. Μπορεί κανείς να διακρίνει τις λακκούβες της οδού στην αριστερή πλευρά του δρόμου.

Εικόνα 14 - Φειδιππίδου (Πανοραμική)

Στην Εικόνα 15 μπορούμε να δούμε από μία πιο κοντινή λήψη την περιοχή ανωμαλιών στο οδόστρωμα της οδού Φειδιππίδου. Πρόκειται για μία σειρά από σπασίματα στο οδόστρωμα καθώς επίσης και μία μικρή λακκούβα από ένα καπάκι αποχέτευσης.

Εικόνα 15 - Φειδιππίδου (Περιοχή ανωμαλιών οδοστρώματος)

Τα οχήματα που κινούνται κατά μήκος της οδού περνάνε επάνω από αυτά τα χαλασμένα τμήματα του δρόμου όπως φαίνεται και στην Εικόνα 16.

Εικόνα 16 - Φειδιππίδου (Κινούμενα οχήματα)

Η καταγραφή της εφαρμογής Safe-R για αυτή την οδό είναι η εξής:

Ονομασία	**Φειδιππίδου**
Μήκος	17 μ
Μέση επιτάχυνση (avgForce)	10.00
Μέγιστη επιτάχυνση (maxForce)	11.22
Διασπορά επιτάχυνσης (varForce)	0.62
Μέση ταχύτητα (avgSpeed)	24.25 km/h
Σύνολο ενημερώσεων (updateCounter)	308
Χαρακτηρισμός οδού	Κακή (κόκκινο χρώμα)

Οι τιμές οι οποίες καταγράφονται υποδηλώνουν αρκετά έντονη αναταραχή με ένα μέγιστο μέτρο επιταχύνσεων μεγέθους 11.22 και μία μέση τιμή μεγέθους 10.00. Το όριο που έχει οριστεί μετά από τα πειράματα για τον κακό δρόμο είναι το 10.7 στη μέγιστη τιμή των επιταχύνσεων και το 10 στο μέση τιμή τους. Εφόσον η συγκεκριμένη οδός ξεπερνάει και τα δύο αυτά όρια, χαρακτηρίζεται ως κακή και έχει κόκκινο χρώμα.

5.2 Οδός Θηβών

Ακολουθεί η περιγραφή και ο σχολιασμός των δύο τμημάτων της οδού Θηβών. Η συγκεκριμένη οδός αποτελείται από ένα μέτριο και ένα κακό τμήμα.

5.2.1 Πρώτο τμήμα οδού Θηβών (μέτρια ποιότητα)

Το πρώτο τμήμα της οδού Θηβών, το οποίο αρχίζει από το τέλος της οδού Φειδιππίδου και φτάνει έως την οδό Μικράς Ασίας, φαίνεται στην Εικόνα 17.

Εικόνα 17 - Οδός Θηβών (Από Φειδιππίδου έως Μικράς Ασίας)

Παρατηρούμε ότι στο συγκεκριμένο τμήμα της οδού υπάρχουν μικρές λακκούβες και σπασίματα κατά μήκος του αλλά σε καμία περίπτωση δε συγκρίνονται σε μέγεθος με αυτές της Φειδιππίδου, όπως αυτή περιγράφηκε στην προηγούμενη ενότητα. Σε αυτή την περίπτωση τα οχήματα έχουν τις ανωμαλίες του οδοστρώματος στη δεξιά μεριά της ροής του δρόμου καθώς κινούνται, παρόλα αυτά όμως τα μέτρα της επιτάχυνσης και πάλι δίνουν πιστά την εικόνα που παρουσιάζει ο δρόμος στα δεδομένα καταγραφής.

Ο δρόμος χαρακτηρίστηκε από την εφαρμογή ως μέτριας ποιότητας (κίτρινο χρώμα) καθώς οι μετρήσεις σε αυτό το τμήμα είναι οι ακόλουθες:

Ονομασία	**Θηβών (Πρώτο τμήμα)**
Μήκος	14.7 μ
Μέση επιτάχυνση (avgForce)	9.83
Μέγιστη επιτάχυνση (maxForce)	10.43
Διασπορά επιτάχυνσης (varForce)	0.12
Μέση ταχύτητα (avgSpeed)	21.86 km/h
Σύνολο ενημερώσεων (updateCounter)	927
Χαρακτηρισμός οδού	Μέτρια (κίτρινο χρώμα)

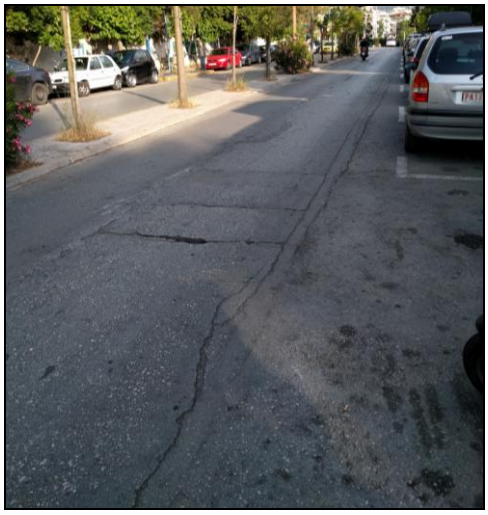
Τα όρια στη μέση και μέγιστη τιμή των επιταχύνσεων έτσι ώστε να χαρακτηριστεί μέτριας ποιότητας ένας δρόμος είναι 9.7 και 10.4 αντίστοιχα. Οπότε οι τιμές της

επιτάχυνσης για τη συγκεκριμένη οδό την κατηγοριοποιούν ως μέτρια. Αξίζει να σημειωθεί εδώ και ο αυξημένος αριθμός μετρήσεων (Snapshots - updateCounter) για τη συγκεκριμένη οδό, οι ο οποίος ανέρχεται στις 927 μετρήσεις. Το γεγονός ότι μετά από τόσες μετρήσεις έχουν προκύψει δεδομένα τα οποία βρίσκονται κοντά στην πραγματική κατάσταση του δρόμου ενισχύει την πιθανότητα να έχουμε πραγματοποιήσει αντικειμενικές μετρήσεις καθ' όλη τη διαδικασία των πειραμάτων καθώς επίσης και το γεγονός ότι οι μετρήσεις συνάδουν με την κατάσταση του δρόμου.

5.2.2 Δεύτερο τμήμα οδού Θηβών (κακή ποιότητα)

Το δεύτερο τμήμα της οδού Θηβών αφορά το ρεύμα που εκτείνεται από την οδό Μικράς Ασίας έως την αρχή της οδού Σπηλιωτοπούλου, στη συμβολή με την οδό Παπαδιαμαντοπούλου. Η συνέχεια αυτή της οδού Θηβών έχει αρκετά χειρότερη εικόνα κατά μήκος της και στο τέλος της στη συμβολή της με την οδό Παπαδιαμαντοπούλου.

Αρχικά, στην Εικόνα 18 παρατηρούμε μερικές εκδορές στο οδόστρωμα και κενά στο πλάτος του. Κάποιες καταγραφές (Snapshot) αναταραχών προέρχονται από αυτά ακριβώς τα σημεία.



Εικόνα 18 - Οδός Θηβών (Από Μικράς Ασίας έως Παπαδιαμαντοπούλου)

Στη συνέχεια, στην Εικόνα 19, υπάρχει και το χειρότερο κομμάτι αυτού του τμήματος με πολλές εναλλαγές στο ύψος του οδοστρώματος που προκαλούν αρκετές αναταράξεις στο όχημα καταγραφής του δρόμου. Αυτό το τμήμα του δρόμου ανήκει κυρίως στην οδό Θηβών και όχι στην οδό Σπηλιωτοπούλου, η οποία εξετάζεται στην επόμενη ενότητα.

Εικόνα 19 - Οδός Θηβών & Παπαδιαμαντοπούλου

Αποτέλεσμα όλων αυτών των κακοτεχνιών στο τμήμα αυτό είναι ο δρόμος να χαρακτηριστεί ως κακής ποιότητας και η συσκευή καταγραφής να έχει συλλέξει τα παρακάτω δεδομένα:

Ονομασία	**Θηβών (Δεύτερο τμήμα)**
Μήκος	10.25 μ
Μέση επιτάχυνση (avgForce)	9.92
Μέγιστη επιτάχυνση (maxForce)	10.95
Διασπορά επιτάχυνσης (varForce)	0.32
Μέση ταχύτητα (avgSpeed)	22.23 km/h
Σύνολο ενημερώσεων (updateCounter)	565
Χαρακτηρισμός οδού	Κακή (κόκκινο χρώμα)

Στην Εικόνα 20 υπάρχει η εικόνα από το χάρτη της εφαρμογής με το δεύτερο τμήμα της Θηβών. Το τμήμα αυτό έχει χαρακτηριστεί με το χαρακτηριστικό κόκκινο χρώμα.

Εικόνα 20 - Οδοί Θηβών & Σπηλιωτοπούλου

Στην ίδια εικόνα μπορούμε να παρατηρήσουμε και την οδό Σπηλιωτοπούλου (με το μπλε χρώμα στο χάρτη) η οποία έχει χαρακτηριστεί ως καλής ποιότητας. Ο σχολιασμός της οδού αυτής ακολουθεί στην επόμενη ενότητα.

5.3 Οδός Σπηλιωτοπούλου (καλή ποιότητα)

Σε αυτή την ενότητα θα μελετήσουμε την οδό Σπηλιωτοπούλου, στο τμήμα από την συμβολή της με την οδό Παπαδιαμαντοπούλου έως και την οδό Δίκης. Αυτά τα δύο τμήματα, που χωρίζονται από την οδό Αγίας Λαύρας, είναι εξίσου ομαλά και χαρακτηρίζονται και τα δύο ως καλής ποιότητας από την εφαρμογή.

Οι δύο εικόνες παρακάτω (Εικόνα 21 και Εικόνα 22) παρουσιάζουν μία εμφανώς καλύτερη εικόνα οδού από ότι οι προηγούμενες οδοί που παρουσιάστηκαν. Δεν υπάρχουν εμφανείς λακκούβες ή κακοτεχνίες γενικότερα. Δεν εμφανίζονται αλλαγές στο ύψος των τμημάτων του δρόμου οπότε οι αναταραχές που καταγράφονται είναι αρκετά μικρές. Η γενικότερη κατάσταση της οδού υποδηλώνει μία αρκετά νεότερη κατασκευή συγκριτικά με τις δύο προηγούμενες οδούς που περιγράφηκαν.

Εικόνα 21 - Οδός Σπηλιωτοπούλου (Από Παπαδιαμαντοπούλου έως Αγίας Λαύρας)

Εικόνα 22 - Οδός Σπηλιωτοπούλου (Από Αγίας Λαύρας έως Δίκης)

Όπως ήταν αναμενόμενο οι μετρήσεις της συσκευής καταγραφής αποτυπώνουν αυτή την καλή ποιότητα. Παρακάτω παρουσιάζονται τα δεδομένα που έχουν καταγραφεί:

Ονομασία	**Σπηλιωτοπούλου**
Μήκος	16.91 μ
Μέση επιτάχυνση (avgForce)	9.87
Μέγιστη επιτάχυνση (maxForce)	10.18
Διασπορά επιτάχυνσης (varForce)	0.03
Μέση ταχύτητα (avgSpeed)	25.73 km/h
Σύνολο ενημερώσεων (updateCounter)	928
Χαρακτηρισμός οδού	Καλή (μπλε χρώμα)

Για ακόμη μία φορά παρατηρούμε ότι οι μετρήσεις είναι πολλές (928 μετρήσεις) δίνοντας το τελικό αποτέλεσμα πολύ κοντά στην πραγματική εικόνα του δρόμου. Τα δεδομένα επιτάχυνσης παρατηρούνται σε χαμηλά επίπεδα οπότε και η οδός χαρακτηρίστηκε από την εφαρμογή ως καλής ποιότητας.

5.4 Συμπεράσματα

Από τα πειράματα που παρουσιάστηκαν στις προηγούμενες ενότητες θα μπορούσαμε να θεωρήσουμε ότι η εφαρμογή είναι σε θέση να κάνει μία εκτίμηση και κατηγοριοποίηση των οδών στις τρεις προκαθορισμένες κατηγορίες. Επομένως να καταλήξουμε στο συμπέρασμα ότι μία Android συσκευή, με χρήση του κατάλληλου λογισμικού, είναι σε θέση να δώσει μία εκτίμηση για το οδόστρωμα στο οποίο κινείται ένα όχημα καταγραφής. Η ακρίβεια των αποτελεσμάτων προφανώς αποτελεί αντικείμενο έρευνας και πολύ λεπτομερέστερης προσέγγισης στα δεδομένα επιτάχυνσης, όπως αυτά συλλέγονται από το επιταχυνσιόμετρο μίας συσκευής Android.

Πολλά προβλήματα που παρουσιάζει η λογική της συλλογής όπως είναι η ποικιλομορφία σε συσκευές, οχήματα αλλά και θέσεις τοποθέτησης της συσκευής αντιμετωπίστηκαν με τη σύμβαση της μοναδικότητας στις τρεις αυτές παραμέτρους. Η λογική που εισάγεται με την είσοδο όλων των πιθανών συσκευών, οχημάτων ή ακόμη και θέσεων τοποθέτησης ξεπερνάει τα πλαίσια της εργασίας και αποτελεί αντικείμενο περεταίρω έρευνας και μοντελοποίησης. Εξάλλου, μία τέτοιου είδους εξέλιξη της εφαρμογής προτείνεται και στο επόμενο κεφάλαιο που αφορά το μέλλον και τα πιθανά σενάρια εξέλιξης της έρευνας και της εφαρμογής.

Δεδομένης, όμως, της αντικειμενικότητας που τηρήθηκε κατά τη συλλογή των δεδομένων και της πραγματικής εικόνας των δρόμων, όπως αυτή παρατηρείται μέσα από το φωτογραφικό υλικό, μπορούμε να πούμε ότι υπάρχει μία εμφανής συνάφεια των τελικών αποτελεσμάτων του χάρτη της εφαρμογής Safe-R και κατ' επέκταση των δεδομένων της ΒΔ, με την ποιότητα του οδοστρώματος.

Στο επόμενο κεφάλαιο γίνεται μία αναφορά σε διάφορα σενάρια επέκτασης και εμπλουτισμού της εφαρμογής. Γίνεται, επίσης, και αναφορά για την ανάγκη περαιτέρω έρευνας επάνω στη μοντελοποίηση των αισθητήρων των συσκευών Android ώστε να υπάρχει η δυνατότητα περισσότερων και πιο ακριβέστερων δεδομένων στο μέλλον.

5.4.1 Δεδομένα ατυχημάτων

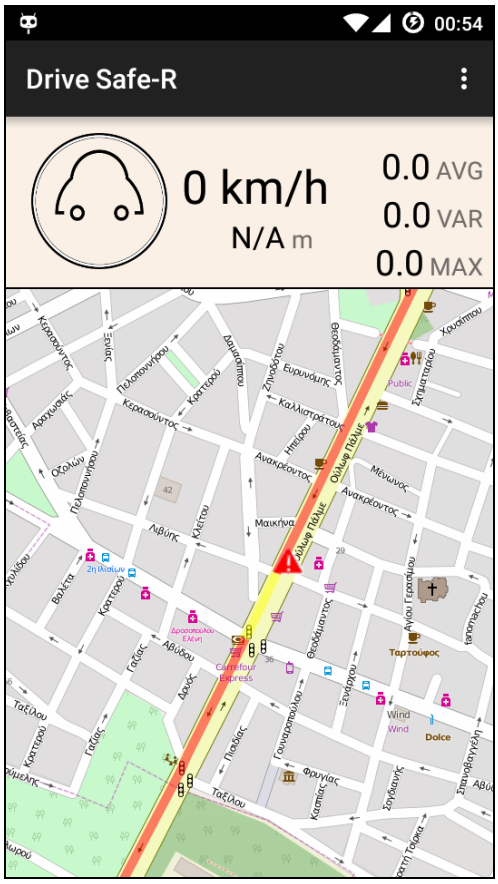
Μετά το πέρας της συλλογής των δειγμάτων από τους δρόμους της περιοχής δοκιμών έγινε μία προσπάθεια σύγκρισης των εκτιμήσεων με πραγματικά δεδομένα ατυχημάτων, όπως αυτά βρίσκονται καταγεγραμμένα στον ηλεκτρονικό χάρτη της ιστοσελίδας http://crashmap.okfn.gr/.

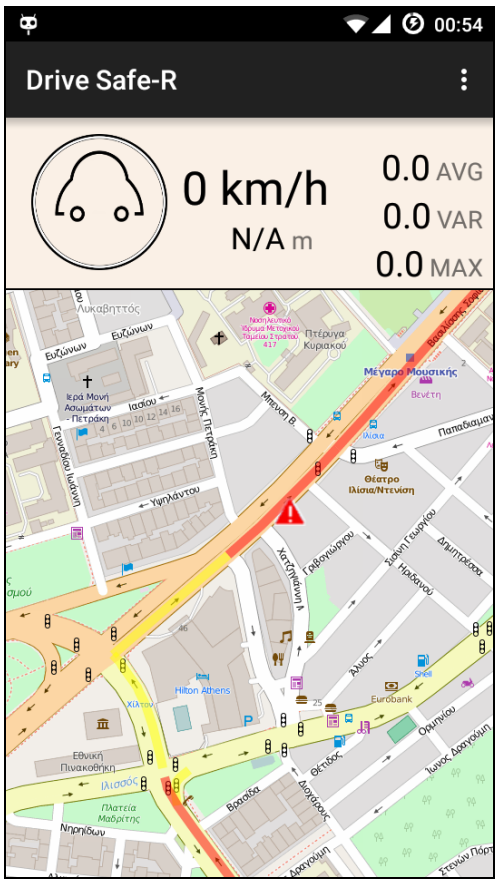
Η ιστοσελίδα okfn.gr αποτελεί το ελληνικό παράρτημα του οργανισμού Open Knowledge Foundation. Το Open Knowledge Foundation είναι ένας μη-κερδοσκοπικός οργανισμός που ιδρύθηκε το 2004 και είναι αφιερωμένος στην προώθηση των ανοιχτών δεδομένων και γενικότερα του ανοιχτού περιεχομένου, συμπεριλαμβανομένων των δεδομένων της κυβέρνησης. Για τη δημιουργία της ιστοσελίδας http://crashmap.okfn.gr/ συνδυάστηκαν δεδομένα που δημοσιεύτηκαν από δημόσιες υπηρεσίες μαζί με δεδομένα που παράχθηκαν από πολίτες και προβάλλονται πάνω σε ένα χάρτη μαζί με διάφορα στατιστικά στοιχεία.

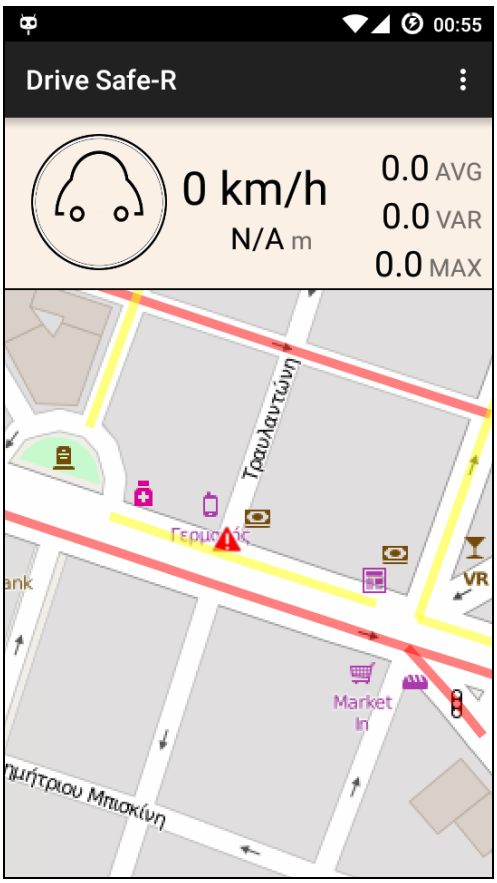
Επιλέχθηκαν σημεία τα οποία βρίσκονται εντός του χάρτη δοκιμών και είναι πιθανό να χαρακτηρίζουν ατύχημα το οποίο έχει ως αιτία την ποιότητα του οδοστρώματος. Συγκεκριμένα επιλέχθηκαν ατυχήματα που αφορούν ανατροπές δικύκλων. Έπειτα τα σημεία αυτά αναπαραστάθηκαν στο χάρτη της εφαρμογής Safe-R σε μία προσπάθεια συσχέτισης της ποιότητας του δρόμου με τα ατυχήματα που έχουν συμβεί στον κάθε δρόμο.

Το δείγμα ήταν αρκετά μικρό, μόλις τέσσερα ατυχήματα που αφορούν ανατροπή μηχανής υπήρχαν καταγεγραμμένα στην εν λόγω ιστοσελίδα. Στις εικόνες που

ακολουθούν φαίνεται η καταγραφή της συσκευής στους δρόμους τους οποίος έχουν καταγραφεί τα ατυχήματα.







Η βασικότερη παρατήρηση που μπορεί να γίνει κοιτώντας αυτό το πολύ μικρό δείγμα εικόνων είναι ότι όλα τα ατυχήματα συνέβησαν σε δρόμο είτε κακής είτε μέτριας κατάστασης, σύμφωνα πάντα με τα δεδομένα καταγραφής της εφαρμογής Safe-R. Προφανώς θα ήταν τελείως αυθαίρετο να συνδέσει κανείς την ποιότητα του οδοστρώματος με τα συγκεκριμένα ατυχήματα δεδομένου ότι η βασική αιτία του δεν είναι καταγεγραμμένη.

Παρόλα αυτά, όμως, γίνεται αυτή η αντιπαράθεση των δεδομένων σαν ένα ακόμη στοιχείο προς παρατήρηση επιπρόσθετα σε όλα τα προηγούμενα. Μία περαιτέρω μελέτη σε πιθανό συσχετισμό της ποιότητας του οδοστρώματος και της συχνότητας των ατυχημάτων είναι ζητούμενη και επιθυμητή. Μάλιστα, δεδομένης και της περιοχής δοκιμών θα ήταν πιο λογική η μελέτη συσχετισμού ατυχημάτων - ποιότητας οδοστρώματος σε διαφορετική περιοχή, η οποία να μη βρίσκεται σε κάποια αστική ζώνη αλλά σε επαρχιακό οδικό δίκτυο. Σε ένα επαρχιακό οδικό δίκτυο είναι λογικό τα ατυχήματα που αφορούν την ποιότητα του οδοστρώματος να είναι μεγαλύτερα σε ποσοστό από ότι εντός μίας πόλης, όπου τα ατυχήματα μπορεί να οφείλονται στον αυξημένο αριθμό οχημάτων.

6. Εξέλιξη Εφαρμογής

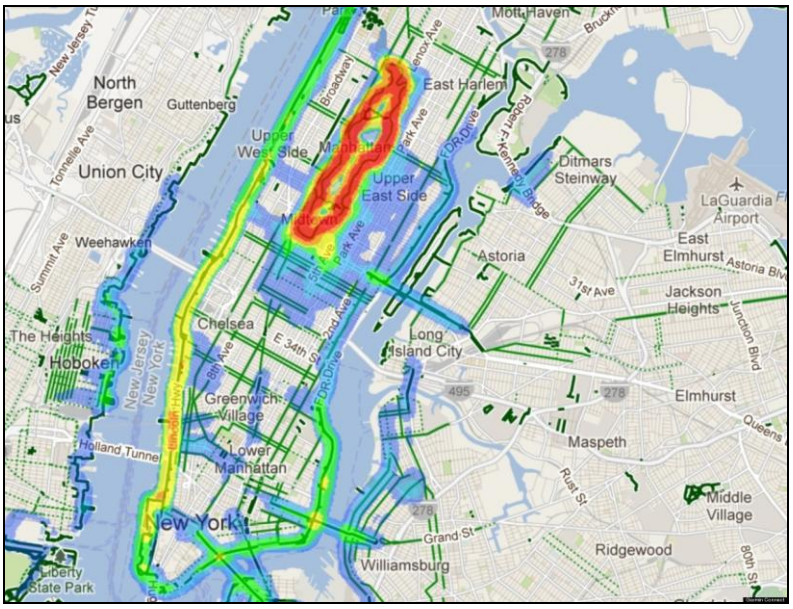
Υπάρχουν αρκετά πεδία στα οποία υπάρχει η δυνατότητα εξέλιξης για την εφαρμογή Safe-R. Παρακάτω παρουσιάζονται πέντε διαφορετικοί τομείς οι οποίοι θα μπορούσαν να αποτελέσουν αντικείμενο εργασίας και έρευνας σε μελλοντικούς ερευνητές.

6.1 Αυξημένη κατηγοριοποίηση

Κατά τη διαδικασία της υλοποίησης έγινε η επιλογή τριών διαφορετικών κατηγοριών δρόμων έτσι ώστε να κατηγοριοποιηθούν όλοι οι δρόμοι εντός της περιοχής δοκιμών. Το γεγονός αυτό μπορεί να οδηγήσει σε ομαδοποίηση αρκετά διαφορετικών δρόμων. Θα μπορούσε να γίνει επιλογή πολύ περισσότερων κατηγοριών, επομένως και χρωμάτων, για την ομαδοποίηση των δρόμων.

Επίσης, θα ήταν αρκετά χρήσιμη η παρουσίαση των καταγεγραμμένων δεδομένων σε μορφή heat-map όπου δεν ορίζονται συγκεκριμένες κατηγορίες χρωμάτων αλλά υπάρχει ένα εύρος χρωμάτων και κάθε διαφορετική τιμή δέχεται το αντίστοιχο αναλογικό τόνο χρωματισμού. Ένα παράδειγμα μίας τέτοιας υλοποίησης θα θύμιζε την Εικόνα 23 όπου φαίνονται οι οδοί της Νέας Υόρκης με βάση τη συχνότητα με την οποία τις χρησιμοποιούν οι δρομείς.

Το αποτέλεσμα στην περίπτωση του Safe-R θα έδινε μία πιο σαφή εικόνα για το μέγεθος της διαφοράς στην ποιότητα μεταξύ των δρόμων σε μία περιοχή.



Εικόνα 23 - Παράδειγμα heat map

6.2 Μοντελοποίηση περισσότερων συσκευών - οχημάτων

Από το πρώτο κεφάλαιο της παρούσας εργασίας έγινε σαφές ότι η μοντελοποίηση πολλών διαφορετικών συσκευών Android ή ακόμη και οχημάτων αποτελεί αντικείμενο περαιτέρω έρευνας. Η διαδικασία αυτή θα απαιτούσε δοκιμές με πολλές διαφορετικές συσκευές και οχήματα. Είναι μια χρονοβόρα διαδικασία και ίσως και σε ένα βαθμό ανούσια δεδομένης της διαρκούς εξέλιξης των συσκευών. Ίσως η ιδανική προσέγγιση σε μία τέτοιου είδους έρευνα θα ήταν η επιλογή μόνο συγκεκριμένων συσκευών. Αυτές οι συσκευές θα μπορούσαν να είναι οι πιο ευρέως χρησιμοποιούμενες. Μία αντίστοιχη λογική θα έπρεπε να εφαρμοστεί και στα περισσότερο χρησιμοποιούμενα οχήματα.

Άμεσο αποτέλεσμα μίας τέτοιας διεργασίας θα ήταν η δυνατότητα συλλογής δεδομένων από πολύ περισσότερες συσκευές και χρήστες. Οπότε, η δημιουργία και ο

εμπλουτισμός του χάρτη θα μπορούσε να γίνει πιο γρήγορος και ευρύτερος σε περιοχή κάλυψης. Τέλος, ακόμη και οι πιο πρόσφατες αλλαγές στο οδόστρωμα θα ήταν αμεσότερα διαθέσιμες στο ευρύ κοινό.

6.3 Εντοπισμός - Παρουσίαση Λακκούβας

Σε όλη την υλοποίηση της εφαρμογής έγινε χρήση των ήδη προ-εγγεγραμμένων στη ΒΔ δρόμων από τον οργανισμό OpenStreetMap. Τα οδικά τμήματα αυτά χαρακτηρίζονται ως ολότητες (όλο το μήκος τους) από τις εγγραφές (Snapshots), της συσκευής, που βρίσκονται εντός τους. Αποτέλεσμα αυτής της διαδικασίας είναι ένα δρόμος με καλά και κακά τμήματα να χαρακτηρίζεται εξ ολοκλήρου ως μέτριος ή κακός παρά το γεγονός ότι μόνο σε κάποια τμήματά του εμφανίζονται ανωμαλίες.

Εάν γίνει ομαδοποίηση των δεδομένων (Snapshots) γύρω από τα σημεία εκείνα τα οποία εμφανίζουν τις μέγιστες τιμές της επιτάχυνσης θα μπορούσαμε να εντοπίσουμε με ακρίβεια τις λακκούβες στο οδικό δίκτυο. Επομένως, ως αποτέλεσμα θα είχαμε μία απεικόνιση όχι απλά μίας γενικότερης εικόνας του οδικού δικτύου αλλά μίας χαρτογράφησης συγκεκριμένων σημείων που χρειάζονται την παρέμβαση των αρμόδιων υπηρεσιών ώστε να βελτιωθούν ή να επιδιορθωθούν πιο στοχευμένα.

6.4 Δρομολόγηση με βάση την ποιότητα του δρόμου

Έχοντας τα δεδομένα των αναταραχών στη ΒΔ υπάρχει η δυνατότητα χρήσης τους ώστε να υπάρξει η δυνατότητα δρομολόγησης με βάση την ποιότητα του οδοστρώματος. Υπάρχουν έτοιμα πακέτα προγραμματισμού ώστε να επιτρέπουν τη χρήση δεδομένων δρόμων του OpenStreetMap σε συνεργασία με δεδομένα άλλου τύπου, όπως τα δεδομένα αναταραχών στην περίπτωσή μας, και να δώσουν τη δυνατότητα δρομολόγησης συνυπολογίζοντας και αυτά. Ένα παράδειγμα τέτοιου πακέτου αποτελεί το Routino[2]. Αλλά υπάρχουν και αρκετά παρόμοια πακέτα ανοικτού κώδικα.

6.5 Επαλήθευση δεδομένων

Η χρήση της εφαρμογής Safe-R, όπως αυτή έχει περιγραφεί, εκτός της αρχικής ενεργοποίησης της καταγραφής από τον χρήστη δεν έχει κάποια άλλη δυνατότητα επικοινωνίας με τον χρήστη.

Θα μπορούσε να δημιουργηθεί μία διεπαφή επικοινωνίας με τον χρήστη η οποία θα ήταν διαθέσιμη στο χρήστη που πραγματοποιεί μία καταγραφή και θα μπορούσε να δώσει τη δυνατότητα στον χρήστη να δηλώσει και ο ίδιος αλλαγές στην ποιότητα του οδοστρώματος. Έτσι, εάν για παράδειγμα υπάρχει μία λακκούβα σε έναν δρόμο η οποία έχει επιδιορθωθεί θα μπορούσαν οι διαφορετικές μετρήσεις της εφαρμογής σε συνδυασμό με την παροχή πληροφοριών των χρηστών να προκαλέσουν την επανεκκίνηση των μετρήσεων για έναν δρόμο με μηδενισμό των υπαρχόντων δεδομένων.

Τέλος, μία ακόμη διεπαφή θα μπορούσε να δημιουργηθεί για τους υπεύθυνους των αρμόδιων υπηρεσιών συντήρησης του οδικού δικτύου. Έτσι σε περίπτωση κατασκευής κάποιου δρόμου θα μπορούσε να είναι άμεση αυτή η επαλήθευση και ο μηδενισμός των δεδομένων του δρόμου ώστε να ξεκινήσει εκ νέου η συλλογή και καταγραφή δεδομένων για αυτόν.

[2] Πηγή: http://www.routino.org/

6.6 Συσχετισμός περισσότερων ατυχημάτων

Στα πλαίσια της παρούσας εργασίας έγινε μία πολύ απλή προσέγγιση στο συσχετισμό ατυχημάτων με την ποιότητα του οδικού δικτύου. Το μέγεθος της περιοχής δοκιμών σε συνδυασμό με τα λίγα ελεύθερα δεδομένα σχετικά με τα ατυχήματα οδήγησαν σε ένα πολύ μικρό μέγεθος χρήσιμων δεδομένων (τέσσερα ατυχήματα εντός της περιοχής δοκιμών).

Επομένως, δεδομένου ότι η περιοχή δοκιμών αυξάνεται σε μέγεθος και πλήθος δεδομένων καθώς επίσης ότι γίνεται μία σε βάθος έρευνα στα τροχαία ατυχήματα εντός της Ελληνικής επικράτειας, θα μπορούσε να γίνει μία ευρύτερη και περισσότερο εμπεριστατωμένη έρευνα του συσχετισμού ατυχημάτων και ποιότητας οδοστρώματος.

Προφανώς, υπάρχουν αρκετές αιτίες πέραν της ποιότητας του δρόμου, οι οποίες είναι οι βασικοί λόγοι για τους οποίους παρατηρούμε τροχαία ατυχήματα, αλλά μία προσέγγιση ακόμη και μόνο σε έναν από τους παράγοντες της συνάρτησης θα έδινε χρήσιμα συμπεράσματα. Μάλιστα η προβολή όχι μόνο της ποιότητας ολόκληρων τμημάτων δρόμων αλλά ο εντοπισμός της λακκούβας θα έδινε με περισσότερη ακρίβεια τις επικίνδυνες θέσεις επάνω στο οδόστρωμα και εάν τελικά ήταν οι λακκούβες ο βασικός λόγος για τον οποίο προκλήθηκαν τα εκάστοτε ατυχήματα.

ΠΙΝΑΚΑΣ ΟΡΟΛΟΓΙΑΣ

Ξενόγλωσσος όρος	**Ελληνικός Όρος**
Snapshot	Στιγμιότυπο
Frame	Πλαίσιο
Server	Εξυπηρετητής

ΣΥΝΤΜΗΣΕΙΣ – ΑΡΚΤΙΚΟΛΕΞΑ – ΑΚΡΩΝΥΜΙΑ

Safe-R	Safe-Road
OSM	OpenStreetMap
ΒΔ	Βάση Δεδομένων
JSON	JavaScript Object Notation
HTTP	Hypertext Transfer Protocol
pbf	Protocolbuffer Binary Format
XML	Extensible Markup Language

ΠΑΡΑΡΤΗΜΑ

SaferAccelerometer.java

```
public class SaferAccelerometer implements SensorEventListener {

    public static final int SAMPLES_DECIMAL_PLACES_PRECISION = 3;

    private int samplesCount = 0;
    private float x, y, z;
    private double max, average, variance, lastSample;
    private boolean isSensorStarted, isSamplingSessionStarted;

    public SaferAccelerometer(Context context) {
        SensorManager sensorManager = (SensorManager)
context.getSystemService(Context.SENSOR_SERVICE);
        sensorManager.registerListener(
                this, sensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER),
                SensorManager.SENSOR_DELAY_UI);
    }

    public void startSensor() {
        this.isSensorStarted = true;
    }

    public void startNewSamplingSession() {
        this.isSamplingSessionStarted = true;
        initializeVars();
    }

    public void stopSamplingSession() {
        this.isSamplingSessionStarted = false;
    }

    public double getMax() {
        return SaferMath.getInstance().round(this.max, SAMPLES DECIMAL PLACES PRECISION);
    }

    public double getAverage() {
        return SaferMath.getInstance().round(this.average, SAMPLES_DECIMAL_PLACES_PRECISION);
    }

    public double getVariance() {
        return SaferMath.getInstance().round(this.variance, SAMPLES DECIMAL PLACES PRECISION);
    }

    public void stopSensor() {
        this.isSensorStarted = false;
    }

    @Override
    public void onSensorChanged(SensorEvent event) {
        if (this.isSensorStarted) {
            this.x = event.values[0];
            this.y = event.values[1];
            this.z = event.values[2];

            if (this.isSamplingSessionStarted) {
                this.lastSample = this.getMagnitude();
                calculateVariance();
                calculateAverage();
                calculateMax();
            }
        }
    }

    @Override
    public void onAccuracyChanged(Sensor sensor, int accuracy) {
    }

    private double getMagnitude() {
        return Math.sqrt(
                this.x * this.x
                + this.y * this.y
                + this.z * this.z);
    }

    private void calculateMax() {
```

```
        if (this.lastSample > this.max) {
            this.max = this.lastSample;
        }
    }

    private void calculateVariance() {
        this.samplesCount++;
        if (this.samplesCount > 1) {
            this.variance = ((this.samplesCount - 2) * this.variance / (this.samplesCount - 1))
                    + Math.pow((this.lastSample - this.average), 2) / this.samplesCount;
        }
    }

    private void calculateAverage() {
        this.average = ((this.samplesCount - 1) * this.average + this.lastSample) /
this.samplesCount;
    }

    private void initializeVars() {
        this.max = 0.0f;
        this.average = 0.0f;
        this.variance = 0.0f;
        this.samplesCount = 0;
    }
}
```

SaferLocationManager.java

```
public class SaferLocationManager extends Service implements LocationListener {

    public static final int GPS_LOCATION_DATA_DECIMAL_PLACES_PRECISION = 5;
    public static final double ACCURACY_THRESHOLD_IN_METERS = 13.0;
    public static final double MIN_SPEED_THRESHOLD_IN_KM_PER_HOUR = 10.0;
    public static final double MAX_SPEED_THRESHOLD_IN_KM_PER_HOUR = 100.0;

    //The minimum distance to change updates in meters
    private static final long MIN_DISTANCE_CHANGE_FOR_UPDATES_IN_METERS = 2;

    //The minimum time between updates in milliseconds
    private static final long SAMPLING_RATE_IN_MILLISECONDS = 500;

    Location location;
    double latitude;
    double longitude;
    double speed;
    double accuracy;

    protected LocationManager locationManager;

    public SaferLocationManager(Context context) {
        try {
            locationManager = (LocationManager) context.getSystemService(LOCATION_SERVICE);
            locationManager.requestLocationUpdates(
                    LocationManager.GPS_PROVIDER,
                    SAMPLING_RATE_IN_MILLISECONDS,
                    MIN_DISTANCE_CHANGE_FOR_UPDATES_IN_METERS, this);
        } catch (Exception e) {
            Log.e("Error : Location", "Impossible to connect to LocationManager", e);
        }
    }

    public boolean isAccurate() {
        return this.accuracy > 0 && this.accuracy < ACCURACY_THRESHOLD_IN_METERS;
    }

    public void updateLocation() {
        if (canGetLocation()) {
            location = locationManager.getLastKnownLocation(LocationManager.GPS_PROVIDER);
            updateLocationData();
        }
    }

    public GeoPoint getGeoPoint() {
        return new GeoPoint(getLatitude(), getLongitude());
    }

    public double getLatitude() {
        return SaferMath.getInstance().round(latitude,
GPS_LOCATION_DATA_DECIMAL_PLACES_PRECISION);
    }

    public double getLongitude() {
        return SaferMath.getInstance().round(longitude,
GPS_LOCATION_DATA_DECIMAL_PLACES_PRECISION);
    }

    /**
     * Get speed from m/s to km/h
     */
    public double getSpeed() {
        return speed * 3.6f;
    }

    public String getFormattedSpeed() {
        return new DecimalFormat("## km/h").format(this.getSpeed());
    }

    public double getAccuracy() {
        return accuracy;
    }

    public boolean canGetLocation()
    {
        return locationManager.isProviderEnabled(LocationManager.GPS_PROVIDER);
```

```
    }

    @Override
    public void onLocationChanged(Location location) {
    }

    @Override
    public void onProviderDisabled(String provider) {
    }

    @Override
    public void onProviderEnabled(String provider) {
    }

    @Override
    public void onStatusChanged(String provider, int status, Bundle extras) {
    }

    @Override
    public IBinder onBind(Intent intent) {
        return null;
    }

    private void updateLocationData() {
        if (location != null) {
            latitude = location.getLatitude();
            longitude = location.getLongitude();
            accuracy = location.getAccuracy();
            speed = location.getSpeed();
        }
    }
}
```

SaferSnapshot.java

```
public class SaferSnapshot {
    // SafeHouse API version
    @Since(0.1)

    @SerializedName("maxForce")
    private double maxForce;

    @SerializedName("avgForce")
    private double avgForce;

    @SerializedName("varForce")
    private double varForce;

    @SerializedName("speed")
    private double speed;

    @SerializedName("lat")
    private double latitude;

    @SerializedName("lon")
    private double longitude;

    @SerializedName("accuracy")
    private double accuracy;

    public SaferSnapshot(SaferAccelerometer accelerometer, SaferLocationManager locationManager)
{
        this.avgForce = accelerometer.getAverage();
        this.maxForce = accelerometer.getMax();
        this.varForce = accelerometer.getVariance();
        this.speed = locationManager.getSpeed();
        this.latitude = locationManager.getLatitude();
        this.longitude = locationManager.getLongitude();
        this.accuracy = locationManager.getAccuracy();
    }

    public double getAvgForce() {
        return avgForce;
    }

    public double getMaxForce() {
        return maxForce;
    }

    public double getVarForce() {
        return varForce;
    }

    public double getLatitude() {
        return latitude;
    }

    public double getLongitude() {
        return longitude;
    }

    /**
     * We check if lat, lon, accuracy and speed are valid.
     * We also check if current location is inside the test Area.
     */
    public boolean shouldBeAdded(List<LatLng> testAreaPolygon) {
        return (latitude != 0.0 && longitude != 0.0
                && accuracy > 0 && accuracy < SaferLocationManager.ACCURACY_THRESHOLD_IN_METERS
                && speed > SaferLocationManager.MIN_SPEED_THRESHOLD_IN_KM_PER_HOUR
                && speed < SaferLocationManager.MAX_SPEED_THRESHOLD_IN_KM_PER_HOUR
                && PolyUtil.containsLocation(new LatLng(latitude, longitude),testAreaPolygon,
false)
        );
    }
}
```

SaferFrame.java

```
public class SaferFrame {
    // SafeHouse API version
    @Since(0.1)

    @SerializedName("snapshots")
    private ArrayList<SaferSnapshot> snapshots;

    @SerializedName("frame")
    private SaferFrameConfig config;

    private int validSnapshotsAmount = 0;

    public SaferFrame() {
        snapshots = new ArrayList<>();
        config = new SaferFrameConfig(1, 1, 1, 0);
    }

    public void addSnapshot(SaferSnapshot snapshot, List<LatLng> testAreaPolygon) {
        if (snapshot.shouldBeAdded(testAreaPolygon)) {
            snapshots.add(snapshot);
            validSnapshotsAmount++;
        }
    }

    public SaferFrameConfig getConfig() {
        return config;
    }

    public int getValidSnapshotsAmount() {
        return validSnapshotsAmount;
    }

    public Map<String, String> toMap() {
        final Map<String, String> framesHashMap = new HashMap<>();
        Gson gson = new GsonBuilder().setVersion(0.1).setPrettyPrinting().create();
        framesHashMap.put("data", gson.toJson(this));
        return framesHashMap;
    }
}
```

SafeDatabaseHandler.java

```
public class SaferDatabaseHandler extends SQLiteOpenHelper {

    private static SaferDatabaseHandler instance = null;

    private static final int DATABASE_VERSION = 1;
    private static final String DATABASE_NAME = "offlineDatabase";
    private static final String TABLE_OFFLINE_FRAMES = "offlineFrames";
    private static final String TABLE_OFFLINE_FRAMES_ID_COLUMN = "id";
    private static final String TABLE_OFFLINE_FRAMES_OFFLINE_FRAME_COLUMN = "offlineFrame";
    private static final String TABLE_OFFLINE_FRAMES_DATE_CREATED_COLUMN = "dateCreated";

    public static SaferDatabaseHandler getInstance(Context context) {
        if (instance == null) {
            instance = new SaferDatabaseHandler(context);
        }
        return instance;
    }

    private SaferDatabaseHandler(Context context) {
        super(context, DATABASE_NAME, null, DATABASE_VERSION);
    }

    @Override
    public void onCreate(SQLiteDatabase db) {
        String CREATE_CONTACTS_TABLE = "CREATE TABLE " + TABLE_OFFLINE_FRAMES + "("
                + TABLE_OFFLINE_FRAMES_ID_COLUMN + " INTEGER PRIMARY KEY AUTOINCREMENT,"
                + TABLE_OFFLINE_FRAMES_OFFLINE_FRAME_COLUMN + " BLOB NOT NULL,"
                + TABLE_OFFLINE_FRAMES_DATE_CREATED_COLUMN + " INT NOT NULL)";
        db.execSQL(CREATE_CONTACTS_TABLE);
    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
        db.execSQL("DROP TABLE IF EXISTS " + TABLE_OFFLINE_FRAMES);
        onCreate(db);
    }

    public void addOfflineData(Map<String, String> params) {
        SQLiteDatabase db = this.getWritableDatabase();

        byte[] byteArray = getByteArrayFromParams(params);// Convert Map to byte array
        Long tsLong = System.currentTimeMillis() / 1000;
        String currentTimestamp = tsLong.toString();

        ContentValues values = new ContentValues();
        values.put(TABLE_OFFLINE_FRAMES_OFFLINE_FRAME_COLUMN, byteArray);
        values.put(TABLE_OFFLINE_FRAMES_DATE_CREATED_COLUMN, currentTimestamp);

        db.insert(TABLE_OFFLINE_FRAMES, null, values);
        db.close();

        Safer.longToast("Frame stored locally...\n(" + getUnsentFramesCount() + " Frames in
SQLite)");
    }

    public List<Map<String, String>> getAllUnsentFrames() {
        List<Map<String, String>> paramsList = new ArrayList<>();
        String selectQuery = "SELECT * FROM " + TABLE_OFFLINE_FRAMES;

        SQLiteDatabase db = this.getWritableDatabase();
        Cursor cursor = db.rawQuery(selectQuery, null);

        if (cursor.moveToFirst()) {
            do {
                paramsList.add(getParamsFromByteArray(cursor.getBlob(1)));
            } while (cursor.moveToNext());
        }

        return paramsList;
    }

    public int getUnsentFramesCount() {
        String countQuery = "SELECT  * FROM " + TABLE_OFFLINE_FRAMES;
        SQLiteDatabase db = this.getReadableDatabase();
        Cursor cursor = db.rawQuery(countQuery, null);
        int count = cursor.getCount();
```

```
        cursor.close();

        return count;
    }

    public void deleteAllFrames() {
        SQLiteDatabase db = this.getWritableDatabase();
        db.delete(TABLE_OFFLINE_FRAMES, null, null);
        db.close();
    }

    @NonNull
    private byte[] getByteArrayFromParams(Map<String, String> params) {
        ByteArrayOutputStream byteOut = new ByteArrayOutputStream();
        try {
            ObjectOutputStream out = new ObjectOutputStream(byteOut);
            out.writeObject(params);
        } catch (IOException e) {
            e.printStackTrace();
        }
        return byteOut.toByteArray();
    }

    private Map<String, String> getParamsFromByteArray(byte[] byteArray) {
        ByteArrayInputStream byteIn = new ByteArrayInputStream(byteArray);
        try {
            ObjectInputStream in = new ObjectInputStream(byteIn);
            return (Map<String, String>) in.readObject();
        } catch (ClassNotFoundException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
        return null;
    }
}
```

Βιβλιογραφία

http://www.artima.com/

*Introduction to postgis*2005Refractions Research Inc

*Laravel 5 Essentials*2015BirminghamPackt Publishing Ltd.

*Nginx HTTP Server: Adopt Nginx for Your Web Applications to Make the Most of Your Infrastructure and Serve Pages Faster Than Ever.*2010Packt Publishing Ltd

Open Handset Alliance. (2007 йил 5-Ноέμβριος). *http://www.openhandsetalliance.com/.* Retrieved 2016 йил 23-Φεβρουάριος from http://www.openhandsetalliance.com/: http://www.openhandsetalliance.com/press_110507.html

Shankland, S. (2007 йил 13-Ноέμβριος). *Google's Android parts ways with Java industry group*. Retrieved 2016 йил 4-Μάρτιος from CNET News: http://www.cnet.com/news/googles-android-parts-ways-with-java-industry-group/

World Health Organization. (2015). *Global Status Report On Road Safety 2015.* Geneva: World Health Organization.

Ανδρέου, Χ. Θ. (2008). *Δυνατότητες και Περιορισμοί της Υπερφασματικής Τηλεσκόπησης στην Ανίχνευση Ποιοτικών Χαρακτηριστικών του Οδοστρώματος.* Αθήνα: Εθνικό Μετρόβιο Πολυτεχνείο.